Aalto University School of Science and Technology
Department of Biomedical Engineering and Computational Science (BECS)

Espoo, Finland July 6, 2011

Gaussian processes for Bayesian analysis
User guide for Matlab toolbox GPstuff
Version 3.1

Jarno Vanhatalo, Jaakko Riihimäki,
Jouni Hartikainen, and Aki Vehtari

If you use GPstuff, please use reference:
Jarno Vanhatalo, Jaakko Riihimäki, Jouni Hartikainen and Aki Vehtari (2011). Bayesian Modeling with Gaussian Processes using the MATLAB Toolbox GPstuff, *submitted*.

(This manual is subject to some modifications.)

contact: Aki.Vehtari@tkk.fi

Revision history

July 2010	First printing	For GPstuff 2.0
September 2010	New functionalities: mean functions for FULL GP, derivative observations for squared exponential covariance in FULL GP.	For GPstuff 2.1
April 2011	New syntax.	For GPstuff 3.1

Contents

1 Introduction **1**
- 1.1 Bayesian modeling . . . 1
- 1.2 Gaussian process models . . . 2
  - 1.2.1 Gaussian process prior . . . 3

2 Inference and prediction **7**
- 2.1 Conditional posterior of the latent function . . . 8
  - 2.1.1 The posterior mean and covariance . . . 8
  - 2.1.2 Gaussian observation model . . . 8
  - 2.1.3 Laplace approximation . . . 9
  - 2.1.4 Expectation propagation algorithm . . . 10
  - 2.1.5 Markov chain Monte Carlo . . . 10
- 2.2 Marginal likelihood given parameters . . . 11
- 2.3 Marginalization over parameters . . . 12
  - 2.3.1 Maximum a posterior estimate of parameters . . . 12
  - 2.3.2 Grid integration . . . 13
  - 2.3.3 Monte Carlo integration . . . 13
  - 2.3.4 Central composite design . . . 14

3 GPstuff basics: regression and classification **17**
- 3.1 Gaussian process regression: demo_regression1 . . . 17
  - 3.1.1 Constructing the model . . . 17
  - 3.1.2 MAP estimate for the hyperparameters . . . 19
  - 3.1.3 Marginalization over parameters with grid integration . . . 20
  - 3.1.4 Marginalization over parameters with MCMC . . . 21
- 3.2 Gaussian process classification . . . 21
  - 3.2.1 Constructing the model . . . 22
  - 3.2.2 Inference with Laplace approximation . . . 22
  - 3.2.3 Inference with expectation propagation . . . 24
  - 3.2.4 Inference with MCMC . . . 24

4 Sparse Gaussian processes **27**
- 4.1 Compactly supported covariance functions . . . 27
- 4.2 FIC and PIC sparse approximations . . . 29
- 4.3 DTC, SOR, VAR approximations . . . 32
  - 4.3.1 Comparison of sparse GP models . . . 33
  - 4.3.2 Sparse GP models with non-Gaussian likelihoods . . . 33

5 Model assessment and comparison **35**
5.1 Marginal likelihood . . . 35
5.2 Predictive performance estimates . . . 35
5.3 Cross-validation . . . 36
5.3.1 Leave-one-out cross-validation . . . 36
5.3.2 k-fold cross-validation . . . 36
5.4 DIC . . . 37
5.5 Model assessment demos . . . 38
5.5.1 demo_modelassessment1 . . . 38
5.5.2 demo_modelassessment2 . . . 39

6 Covariance functions **41**
6.1 Neural network covariance function . . . 41
6.2 Additive models . . . 41
6.2.1 Additive models demo: demo_periodic . . . 43
6.2.2 Additive models with sparse approximations . . . 45
6.3 Additive covariance functions with selected variables . . . 46
6.4 Product of covariance functions . . . 47

7 Special observation models **51**
7.1 Robust regression with Student-t likelihood . . . 51
7.1.1 Regression with Student-t distribution . . . 52
7.2 Models for spatial epidemiology . . . 54
7.2.1 Disease mapping with Poisson likelihood: demo_spatial1 . . . 55
7.2.2 Disease mapping with negative Binomial likelihood . . . 56
7.3 Log-Gaussian Cox process . . . 57
7.4 Binomial observation model . . . 58
7.5 Derivative observations in GP regression . . . 59
7.5.1 GP regression with derivatives: demo_derivativeobs . . . 60

8 Mean functions **63**
8.1 Explicit basis functions . . . 63
8.1.1 Mean functions in GPstuff: demo_regression_meanf . . . 65

A Function list **67**
A.1 GP . . . 67
A.2 Diagnostic tools . . . 69
A.3 Distributions . . . 69
A.4 Monte Carlo . . . 70
A.5 Miscellaneous . . . 71
A.6 Optimization . . . 71

B Covariance functions **73**

C Observation models **77**

D Priors **79**

E Transformation of hyperparameters **83**

Acknowledgments

The coding of the GPstuff-toolbox started in 2006 based on the MCMCStuff-toolbox (1998-2006) http://www.lce.hut.fi/research/mm/mcmcstuff/ which was based on Netlab-toolbox (1996-2001) <http://www.mathworks.com/matlabcentral/fileexchange/2654-netlab>.

The GPstuff-toolbox has been developed by BECS Bayes group, Aalto University. The main authors of the GPstuff have been Jarno Vanhatalo, Jaakko Riihim?i, Jouni Hartikainen and Aki Vehtari, but the package contains code written by many more people. In the Bayesian methodology group at the Department of Biomedical Engineering and Computational Science (BECS) at Aalto University these persons are (in alphabetical order): Toni Auranen, Roberto Calandra, Pasi Jyl?ki, Tuomas Nikoskinen, Eero Pennala, Heikki Peura, Ville Pietil?nen, Markus Siivola, Simo S?kk? and Ville Tolvanen. People outside Aalto University are (in alphabetical order): Christopher M. Bishop, Timothy A. Davis, Dirk-Jan Kroon, Ian T. Nabney, Radford M. Neal and Carl E. Rasmussen. We want to thank them all for sharing their code under a free software license.

The research leading to GPstuff was funded by Helsinki University of Technology, Aalto University, the Academy of Finland, Finnish Funding Agency for Technology and Innovation (TEKES), and the Graduate School in Electronics and Telecommunications and Automation (GETA). Jarno Vanhatalo also thanks the Finnish Foundation for Economic and Technology Sciences KAUTE, Finnish Cultural Foundation, and Emil Aaltonen Foundation for supporting his post graduate studies during which the code package was prepared.

Preface

This is a manual for software package GPstuff, which is a collection of Matlab functions to build and analyse Bayesian models build over Gaussian processes. The purpose of the manual is to help people to use the software in their own work and possibly modify and extend the features. The manual consist of two short introductory sections about Bayesian inference and Gaussian processes, which introduce the topic and the notation used throughout the book. The theory is not extensively covered and readers not familiar with it are suggested to see the references for a more complete discussion.

After the introductory part, the Chapter 2 is devoted for inference techniques. Gaussian process models lead to analytically unsolvable models for which reason efficient approximative methods are essential. The techniques implemented in GPstuff are introduced in general level and references are given to direct the reader for more detailed discussion on the methods.

The rest of the manual discusses the basic models implemented in the package and demonstrates their usage. This discussion begins from the Chapter 3 which considers simple Gaussian process regression and classification problems. These examples serve also as examples how to use the basic functions in the GPstuff and all the rest of the examples build over the considerations in this chapter. The rest of the chapters concentrate on more special model constructions, such as sparse Gaussian processes, additive models, and various observations models. Also, functions for model assessment and comparison are discussed. All these considerations are more or less individual examples that can be read if needed. The essential parts that everyone should know are covered by the end of the chapter 3.

Chapter 4 discusses how various sparse approximations can be used with GPstuff. The Chapter 5 discusses methods provided for model assessment and comparison. The Chapter 6 tells more about some covariance functions and how to combine them. The Chapter 7 reviews some special observation models implemented in GPstuff. The Chapter 8 reviews how non-zero mean functios can be used in GPstuff. The Appendix A lists the functions in the toolbox (Contents.m). The Appendices B-D collects technical details and lists of covariance functions, observation models and prior distributions implemented. The Appendix E discusses the reparameterization used and its effect to the implementation.

Chapter 1

Introduction

1.1 Bayesian modeling

Building a Bayesian model, denoted by $\mathcal{M}$, starts with an *observation model* which contains the description of the data generating process, or its approximation. The observation model is denoted by $p(\mathcal{D}|\theta, \mathcal{M})$, where θ stands for the parameters and $\mathcal{D}$ the observations. The observation model quantifies a conditional probability for data given the parameters, and when looked as a function of parameters it is called *likelihood*. If the parameter values were known, the observation model would contain all the knowledge of the phenomenon and could be used as such. If the observations contain randomness, sometimes called *noise*, one would still be uncertain of the future observations, but could not reduce this uncertainty since everything that can be known exactly would be encoded in the observation model. Usually, the parameter values are not known exactly but there is only limited knowledge on their possible values. This prior information is formulated mathematically by the *prior probability* $p(\theta|\mathcal{M})$, which reflects our beliefs and knowledge about the parameter values before observing data. Opposed to the aleatory uncertainty encoded in the observation model the epistemic uncertainty present in the prior can be reduced by gathering more information on the phenomenon (for a more illustrative discussion on the differences between these two sources of uncertainty see (O'Hagan, 2004)). Bayesian inference is the process of updating our prior knowledge based on new observations – in other words it is the process for reducing the epistemic uncertainty.

The cornerstone of Bayesian inference is the Bayes' theorem which defines the conditional probability of the parameters after observing the data

$$p(\theta|\mathcal{D}, \mathcal{M}) = \frac{p(\mathcal{D}|\theta, \mathcal{M})p(\theta|\mathcal{M})}{p(\mathcal{D}|\mathcal{M})}. \tag{1.1}$$

This is called the *posterior distribution* and it contains all information about parameter θ conveyed from the data $\mathcal{D}$ by the model. The normalization constant

$$p(\mathcal{D}|\mathcal{M}) = \int p(\mathcal{D}|\theta, \mathcal{M})p(\theta|\mathcal{M})d\theta \tag{1.2}$$

is equal to the conditional probability that the data came from the model $\mathcal{M}$ given our model assumptions. It is also called the marginal likelihood for the model. The model, $\mathcal{M}$, stands for all the hypotheses and assumptions that are made about the phenomenon. It embodies the functional forms of the observation model and the prior,

which are always tied together, as well as our subjective assumptions used to define these mathematical abstractions. Because everything is conditioned on $\mathcal{M}$, it is a redundant symbol and as such omitted from this on. Usually we are not able to define 'correct' model and most of the time we have only limited ability to encode our prior beliefs in the mathematical formulation. Still, many models turn out to be useful in practice.

The true power of Bayesian approach comes from the possibility to construct and analyze hierarchical models. In hierarchical models, prior probabilities are appointed also for the parameters of the prior. Let us write the prior as $p(\theta|\eta)$, where η denotes the parameters of the prior distribution, *hyperparameters*. By setting a hyperprior, $p(\eta)$, for the hyperparameters we obtain a hierarchical model structure where the fixed parameter values move further away from the data. This allows more flexible models and leads to more vague prior, which is beneficial if the modeler is unsure of the specific form of the prior. In theory the hierarchy could be extended as far as one desires but there are practical limits on how many levels of hierarchy are reasonable (Goel and Degroot, 1981).

The models implemented in GPstuff are hierarchical models where the parameter θ is replaced by a latent function $f(\mathbf{x})$. The observation model is build such that an individual observation depends on a function value at a certain input location $\mathbf{x}$. The latent function $f(\mathbf{x})$ is given a Gaussian process (GP) prior (Rasmussen and Williams, 2006), whose properties are defined by a mean and covariance function, and the hyperparameters related to them. The hierarchy is continued to third level by giving a hyperprior for the covariance function parameters. The assumption is that there is a functional description for the studied phenomenon, which we are not aware of, and the observations are noisy realizations of this underlying function. The power of this construction lies in the flexibility and non-parametric form of the GP prior. We can use simple parametric observation models that describe the assumed observation noise. The assumptions about the functional form of the phenomenon are encoded in the GP prior. Since GP is a non-parametric model we do not need to fix the functional form of the latent function, but we can give implicit statements of it. These statements are encoded in the mean and covariance function, which determine, for example, the smoothness and variability of the function.

1.2 Gaussian process models

The general form of the models in GPstuff can be written as follows:

$$\text{observation model:} \qquad \mathbf{y} \mid \mathbf{f}, \phi \sim \prod_{i=1}^{n} p(y_i|f_i, \phi) \tag{1.3}$$

$$\text{GP prior:} \qquad f(\mathbf{x})|\theta \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'\,|\theta)\right) \tag{1.4}$$

$$\text{hyperprior:} \qquad \theta, \phi \sim p(\theta)p(\phi). \tag{1.5}$$

Here $\mathbf{y} = [y_1, ..., y_n]^{\mathrm{T}}$ is a vector of observations (target values) at (input) locations $\mathbf{X} = \{\mathbf{x}_i = [x_{i,1}, ..., x_{i,d}]^{\mathrm{T}}\}_{i=1}^{n}$. $f(\mathbf{x}) : \Re^d \to \Re$ is a latent function with value $f_i = f(\mathbf{x}_i)$ at input location $\mathbf{x}_i$. A boldface notation will denote a set of latent variables in a vector $\mathbf{f} = [f_1, ..., f_n]^{\mathrm{T}}$. Here, the inputs are real valued vectors but in general other inputs, such as strings or graphs, are possible as well. θ denotes the parameters in the covariance function $k(\mathbf{x}, \mathbf{x}'\,|\theta)$, and ϕ denotes the parameters in the observation model $p(\mathbf{y} \mid \mathbf{f}, \phi)$. The notation will be slightly abused since $p(\mathbf{y} \mid \mathbf{f}, \phi)$ is also used for the

likelihood function. For simplicity of presentation the mean function is considered zero, $m(\mathbf{x}) \equiv 0$, throughout this paper. In Section 8.1.1 we describe how to use non-zero mean functions in GPstuff. As will be seen, the class of models described by the equations (1.3)-(1.5) is rather rich. Even though the observation model is assumed to factorize given the latent variables $\mathbf{f}$, the correlations between the observations are incorporated into the model via the GP prior, and the marginalized observation model $p(\mathbf{y}\,|\phi,\theta) = \int d\,\mathbf{f}\, p(\mathbf{f}\,|\theta) \prod_{i=1}^{n} p(y_i|f_i,\phi)$ is no longer factorizable.

Gaussian processes are not certainly a new invention. Early examples of their usage can be found, for example, in time series analysis and filtering (Wiener, 1949), and geostatistics (e.g. Matheron, 1973). GPs are still widely and actively used in these fields and useful overviews are provided by Cressie (1993), Grewal and Andrews (2001), Diggle and Ribeiro (2007), and Gelfand et al. (2010). O'Hagan (1978) was one of the firsts to consider Gaussian processes in a general probabilistic modeling context. He provided a general theory of Gaussian process prediction and utilized it for a number of regression problems. This general regression framework was later rediscovered as an alternative for neural network models (Williams and Rasmussen, 1996; Rasmussen, 1996), and extended for other problems than regression (Neal, 1997; Williams and Barber, 1998). This machine learning perspective is comprehensively summarized by Rasmussen and Williams (2006).

1.2.1 Gaussian process prior

The advantage of using GPs is that we can conduct the inference directly in the function space. GP prior implies that any set of function values $\mathbf{f}$, indexed by the input coordinates $\mathbf{X}$, have a multivariate Gaussian prior distribution

$$p(\mathbf{f}\,|\mathbf{X},\theta) = \mathrm{N}(\mathbf{f}\,|\mathbf{0},\mathbf{K}_{\mathrm{f,f}}), \tag{1.6}$$

where $\mathbf{K}_{\mathrm{f,f}}$ is the covariance matrix. Notice, that the distribution over functions will be denoted by $\mathcal{GP}(\cdot,\cdot)$, whereas the distribution over a finite set of latent variables will be denoted by $\mathrm{N}(\cdot,\cdot)$. The covariance matrix is constructed from a covariance function, $[\mathbf{K}_{\mathrm{f,f}}]_{i,j} = k(\mathbf{x}_i,\mathbf{x}_j|\theta)$, which characterizes the correlation between different points in the process $\mathrm{E}[f(\mathbf{x}_i), f(\mathbf{x}_j)] = k(\mathbf{x}_i,\mathbf{x}_j|\theta)$. Covariance function encodes the prior assumptions of the latent function, such as the smoothness and scale of the variation, and can be chosen freely as long as the covariance matrices produced are symmetric and positive semi-definite ($\mathbf{v}^{\mathrm{T}}\,\mathbf{K}_{\mathrm{f,f}}\,\mathbf{v} \geq 0, \forall \mathbf{v} \in \Re^n$). An example of a stationary covariance function is the squared exponential

$$k_{\mathrm{se}}(\mathbf{x}_i,\mathbf{x}_j\,|\theta) = \sigma^2_{\mathrm{se}} \exp\left(-\frac{1}{2}\sum_{k=1}^{d}(x_{i,k}-x_{j,k})^2/l_k^2\right), \tag{1.7}$$

where $\theta = [\sigma^2_{\mathrm{se}}, l_1, ..., l_d]$ is a vector of parameters. Here, σ^2_{se} is the scaling parameter, and l_k is the length-scale, which governs how fast the correlation decreases as the distance increases in the direction k. The squared exponential is only one example of possible covariance functions. Discussion on other common covariance functions is given, for example, by (Diggle and Ribeiro, 2007; Finkenstädt et al., 2007; Rasmussen and Williams, 2006). All the covariance functions implemented in GPstuff are summarized in the Chapter 8.

By definition, the marginal distribution of any subset of latent variables can be constructed by simply taking the appropriate submatrix of the covariance and subvector

of the mean. Imagine that we want to predict the values $\tilde{\mathbf{f}}$ at new input locations $\tilde{\mathbf{X}}$. The joint prior for latent variables $\mathbf{f}$ and $\tilde{\mathbf{f}}$ is

$$\begin{bmatrix}\mathbf{f}\\ \tilde{\mathbf{f}}\end{bmatrix} |\mathbf{X}, \tilde{\mathbf{X}}, \theta \sim \mathrm{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{K}_{\mathrm{f,f}} & \mathbf{K}_{\mathrm{f},\tilde{\mathrm{f}}}\\ \mathbf{K}_{\tilde{\mathrm{f}},\mathrm{f}} & \mathbf{K}_{\tilde{\mathrm{f}},\tilde{\mathrm{f}}}\end{bmatrix}\right), \tag{1.8}$$

where $\mathbf{K}_{\mathrm{f,f}} = k(\mathbf{X}, \mathbf{X}|\theta)$, $\mathbf{K}_{\mathrm{f},\tilde{\mathrm{f}}} = k(\mathbf{X}, \tilde{\mathbf{X}}|\theta)$ and $\mathbf{K}_{\tilde{\mathrm{f}},\tilde{\mathrm{f}}} = k(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}|\theta)$. Here, the covariance function $k(\cdot,\cdot)$ denotes also vector and matrix valued functions $k(\mathbf{x}, \mathbf{X}) : \Re^d \times \Re^{d?\times n} \to \Re^{1\times n}$, and $k(\mathbf{X}, \mathbf{X}) : \Re^{d\times n} \times \Re^{d\times n} \to \Re^{n\times n}$. The marginal distribution of $\tilde{\mathbf{f}}$ is $p(\tilde{\mathbf{f}}|\tilde{\mathbf{X}}, \theta) = \mathrm{N}(\tilde{\mathbf{f}}|\mathbf{0}, \mathbf{K}_{\tilde{\mathrm{f}},\tilde{\mathrm{f}}})$ like the marginal distribution of $\mathbf{f}$ given in (1.6). This is illustrated in Figure 1.1. The conditional distribution of a set of latent variables given another set of latent variables is Gaussian as well. For example, the distribution of $\tilde{\mathbf{f}}$ given $\mathbf{f}$ is

$$\tilde{\mathbf{f}}|\, \mathbf{f}, \mathbf{X}, \tilde{\mathbf{X}}, \theta \sim \mathrm{N}(\mathbf{K}_{\tilde{\mathrm{f}},\mathrm{f}}\, \mathbf{K}_{\mathrm{f,f}}^{-1}\, \mathbf{f}, \mathbf{K}_{\tilde{\mathrm{f}},\tilde{\mathrm{f}}} - \mathbf{K}_{\tilde{\mathrm{f}},\mathrm{f}}\, \mathbf{K}_{\mathrm{f,f}}^{-1}\, \mathbf{K}_{\mathrm{f},\tilde{\mathrm{f}}}), \tag{1.9}$$

which can be interpreted as the posterior predictive distribution for $\tilde{\mathbf{f}}$ after observing the function values at locations $\mathbf{X}$. The posterior distribution (1.9) generalizes to a Gaussian process with mean function $m(\tilde{\mathbf{x}}|\theta) = k(\tilde{\mathbf{x}}, \mathbf{X}|\theta)\, \mathbf{K}_{\mathrm{f,f}}^{-1}\, \mathbf{f}$ and covariance $k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}'|\theta) = k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}'|\theta) - k(\tilde{\mathbf{x}}, \mathbf{X}|\theta)\, \mathbf{K}_{\mathrm{f,f}}^{-1}\, k(\mathbf{X}, \tilde{\mathbf{x}}'|\theta)$, which define the posterior distribution of the latent function $f(\tilde{\mathbf{x}})$ on arbitrary input vector $\tilde{\mathbf{x}}$. The posterior GP is illustrated in Figure 1.2.

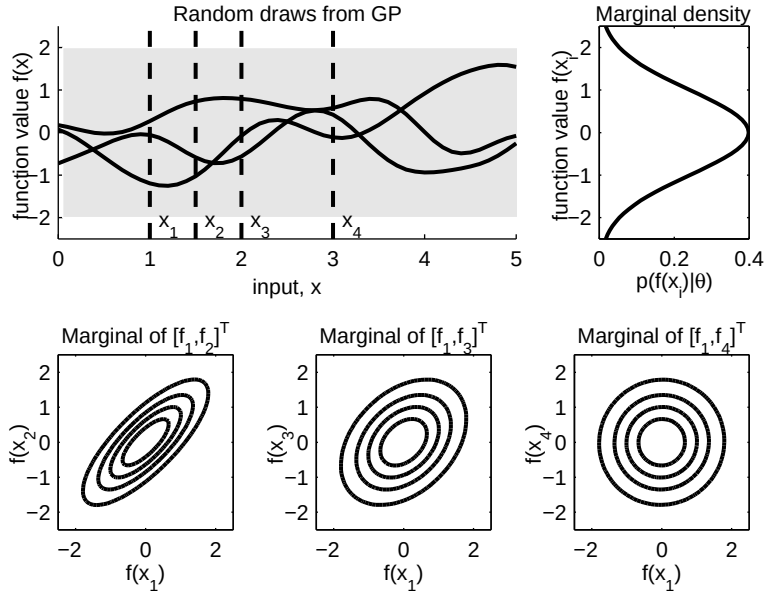


Figure 1.1: An illustration of a Gaussian process. The upper left figure presents three functions drawn randomly from a zero mean GP with squared exponential covariance function. The hyperparameters are $l = 1$ and $\sigma^2 = 1$ and the grey shading represents central 95% probability interval. The upper right subfigure presents the marginal distribution for a single function value. The lower subfigures present three marginal distributions between two function values at distinct input locations shown in the upper left subfigure by dashed line. It can be seen that the correlation between function values $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ is the greater the closer $\mathbf{x}_i$ and $\mathbf{x}_j$ are to each others.

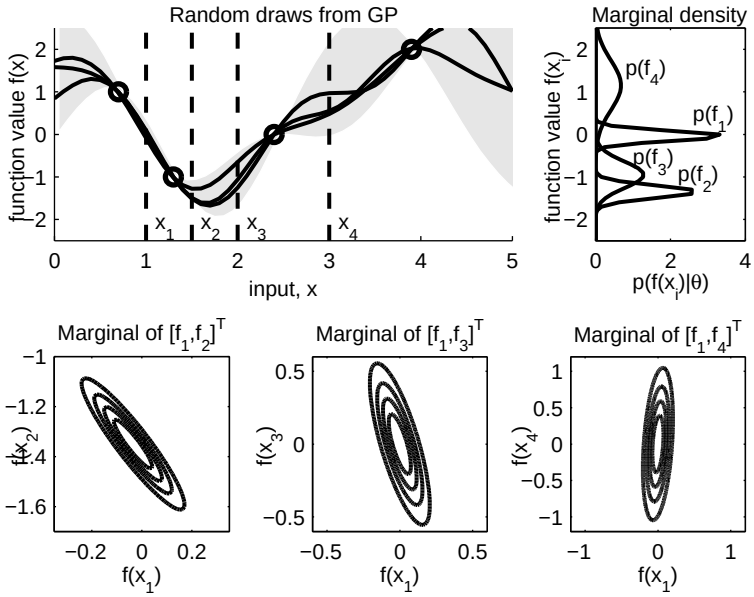


Figure 1.2: A conditional (posterior) GP $p(\tilde{f}|\,\mathbf{f},\theta)$. The observations $\mathbf{f} = [f(0.7) = 1, f(1.3) = -1, f(2.4) = 0, f(3.9) = 2]^{\mathrm{T}}$ are plotted with circles in the upper left subfigure and the prior GP is illustrated in the figure 1.1. When comparing the subfigures to the equivalent ones in Figure 1.1 we can see clear distinction between the marginal and the conditional GP. Here, all the function samples travel through the observations, the mean is no longer zero and the covariance is non-stationary.

Chapter 2

Inference and prediction

As was stated on previous section, the essential task in Bayesian inference is solve the posterior distribution of quantities of interest. In an ideal situation, the posterior distributions could be solved analytically but most of the time we need to approximate it. GPstuff is built so that the first inference step is to form (either analytically or approximately) the conditional posterior of the latent variables given the parameters

$$p(\mathbf{f}\,|\mathcal{D},\theta,\phi) = \frac{p(\mathbf{y}\,|\,\mathbf{f},\phi)p(\mathbf{f}\,|\mathbf{X},\theta)}{\int p(\mathbf{y}\,|\,\mathbf{f},\phi)p(\mathbf{f}\,|\mathbf{X},\theta)d\,\mathbf{f}}, \tag{2.1}$$

where the integral over $\mathbf{f}$ in the denominator is difficult for non-Gaussian likelihoods. Section 2.1 discusses approximation of the posterior of $\mathbf{f}$ first generally and then presents exact posterior in case of Gaussian likelihood and Laplace, Expectation Propagation (EP) and Markov chain Monte Carlo (MCMC) approximations in case of non-Gaussian likelihood. The approximations use the unnormalized posterior

$$p(\mathbf{f}\,|\mathcal{D},\theta,\phi) \propto p(\mathbf{y}\,|\,\mathbf{f},\phi)p(\mathbf{f}\,|,\mathbf{X},\theta), \tag{2.2}$$

and the normalization is computed from the Gaussian approximation itself (Laplace, EP) or the normalization is not needed at all (MCMC).

Section 2.3 treats the problem of marginalizing over the parameters to obtain the marginal posterior distribution for the latent variables

$$p(\mathbf{f}\,|\mathcal{D}) = \int p(\mathbf{f}\,|\mathcal{D},\theta,\phi)p(\theta,\phi|\mathcal{D})d\theta d\phi. \tag{2.3}$$

The posterior predictive distributions can be obtained similarly by first evaluating the conditional posterior predictive distribution, for example $p(\tilde{f}|\mathcal{D},\theta,\phi,\tilde{\mathbf{x}})$, and then marginalizing over the parameters. The joint predictive distribution $p(\tilde{\mathbf{y}}|\mathcal{D},\theta,\phi,\tilde{\mathbf{x}})$ would require integration over possibly high dimensional distribution $p(\tilde{\mathbf{f}}|\mathcal{D},\theta,\phi,\tilde{\mathbf{x}})$ but usually we are interested only on the predictive distribution for each $\tilde{y}_i$ separately. Since the observation model is factorizable this requires only one dimensional integrals (except for multioutput models)

$$p(\tilde{y}_i|\mathcal{D},\tilde{\mathbf{x}}_i,\theta,\phi) = \int p(\tilde{y}_i|\tilde{f}_i,\phi)p(\tilde{f}_i|\mathcal{D},\tilde{\mathbf{x}}_i,\theta,\phi)d\tilde{f}_i. \tag{2.4}$$

For many observation models, which do not have free parameters γ, this integral reduces to marginalization over $\tilde{f}_i$ only.

2.1 Conditional posterior of the latent function

2.1.1 The posterior mean and covariance

If the parameters are considered fixed, GP's marginalization and conditionalization properties can be exploited in the prediction. Assume that we have found the conditional posterior distribution $p(\mathbf{f}\,|\mathcal{D},\theta,\phi)$, which in general, is not Gaussian. We can then evaluate the posterior predictive mean simply by using the expression of the conditional mean $\mathrm{E}_{\tilde{\mathbf{f}}|\,\mathbf{f},\theta,\phi}[f(\tilde{\mathbf{x}})] = k(\tilde{\mathbf{x}},\mathbf{X})\,\mathbf{K}_{\mathrm{f,f}}^{-1}\,\mathbf{f}$ (see equation (1.9) and the text below it). Since this holds for any set of latent variables $\tilde{\mathbf{f}}$, we obtain a parametric posterior mean function

$$m_p(\tilde{\mathbf{x}}|\mathcal{D},\theta,\phi) = \int \mathrm{E}_{\tilde{\mathbf{f}}|\,\mathbf{f},\theta,\phi}[f(\tilde{\mathbf{x}})]p(\mathbf{f}\,|\mathcal{D},\theta,\phi)d\,\mathbf{f} = k(\tilde{\mathbf{x}},\mathbf{X}|\theta)\,\mathbf{K}_{\mathrm{f,f}}^{-1}\,\mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}[\mathbf{f}]. \quad (2.5)$$

The posterior predictive covariance between any set of latent variables, $\tilde{\mathbf{f}}$, can be evaluated from

$$\mathrm{Cov}_{\tilde{\mathbf{f}}|\mathcal{D},\theta,\phi}[\tilde{\mathbf{f}}] = \mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}\left[\mathrm{Cov}_{\tilde{\mathbf{f}}|\,\mathbf{f}}[\tilde{\mathbf{f}}]\right] + \mathrm{Cov}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}\left[\mathrm{E}_{\tilde{\mathbf{f}}|\,\mathbf{f}}[\tilde{\mathbf{f}}]\right], \quad (2.6)$$

where the first term simplifies to the conditional covariance in equation (1.9) and the second term can be written as $k(\tilde{\mathbf{x}},\mathbf{X})\,\mathbf{K}_{\mathrm{f,f}}^{-1}\,\mathrm{Cov}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}[\mathbf{f}]\,\mathbf{K}_{\mathrm{f,f}}^{-1}\,k(\mathbf{X},\tilde{\mathbf{x}}')$. Plugging these in the equation and simplifying gives us the posterior covariance function

$$k_p(\tilde{\mathbf{x}},\tilde{\mathbf{x}}'|\mathcal{D},\theta,\phi) = k(\tilde{\mathbf{x}},\tilde{\mathbf{x}}'|\theta) - k(\tilde{\mathbf{x}},\mathbf{X}|\theta)\left(\mathbf{K}_{\mathrm{f,f}}^{-1} - \mathbf{K}_{\mathrm{f,f}}^{-1}\,\mathrm{Cov}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}[\mathbf{f}]\,\mathbf{K}_{\mathrm{f,f}}^{-1}\right)k(\mathbf{X},\tilde{\mathbf{x}}'|\theta). \quad (2.7)$$

From this on the posterior predictive mean and covariance will be denoted $m_p(\tilde{\mathbf{x}})$ and $k_p(\tilde{\mathbf{x}},\tilde{\mathbf{x}}')$.

Even if the exact posterior $p(\tilde{f}|\mathcal{D},\theta,\phi)$ is not available in closed form, we can still approximate its posterior mean and covariance functions if we can approximate $\mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}$ and $\mathrm{Cov}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}[\mathbf{f}]$. A common practice to approximate the posterior $p(\mathbf{f}\,|\mathcal{D},\theta,\phi)$ is either with MCMC (e.g. Neal, 1997, 1998; Diggle et al., 1998; Kuss and Rasmussen, 2005; Christensen et al., 2006) or by giving an analytic approximation to it (e.g. Williams and Barber, 1998; Gibbs and Mackay, 2000; Minka, 2001; Csató and Opper, 2002; Rue et al., 2009). The analytic approximations considered here assume a Gaussian form in which case it is natural to approximate the predictive distribution with a Gaussian as well. In this case the equations (2.5) and (2.7) give its mean and covariance. The Gaussian approximation can be justified if the conditional posterior is unimodal, which it is, for example, if the likelihood is log concave (this can easily be seen by evaluating the Hessian of the posterior $p(\mathbf{f}\,|\mathcal{D},\theta)$), and there is enough data so that the posterior will be close to Gaussian. A pragmatic justification for using Gaussian approximation is that many times it suffices to approximate well the mean and variance of the latent variables. These on the other hand fully define Gaussian distribution and one can approximate the integrals over $\mathbf{f}$ by using the Gaussian form for their conditional posterior. Detailed considerations on the approximation error and the asymptotic properties of the Gaussian approximation are presented, for example, by Rue et al. (2009) and Vanhatalo et al. (2010).

2.1.2 Gaussian observation model

A special case of an observation model, for which the conditional posterior of the latent variables can be evaluated analytically, is the Gaussian distribution, $y_i \sim \mathrm{N}(f_i,\sigma^2)$,

where the parameter ϕ represents the noise variance σ^2. Since both the likelihood and the prior are Gaussian functions of the latent variable, we can integrate analytically over $\mathbf{f}$ to obtain

$$p(\mathbf{y}\,|\mathbf{X},\theta,\sigma^2) = \mathrm{N}(\mathbf{y}\,|\mathbf{0},\mathbf{K}_{\mathrm{f,f}}+\sigma^2\mathbf{I}). \tag{2.8}$$

Setting this in the denominator of the equation (2.1), gives a Gaussian distribution also for the conditional posterior of the latent variables

$$\mathbf{f}\,|\mathcal{D},\theta,\sigma^2 \sim \mathrm{N}(\mathbf{K}_{\mathrm{f,f}}(\mathbf{K}_{\mathrm{f,f}}+\sigma^2\mathbf{I})^{-1}\,\mathbf{y}, \mathbf{K}_{\mathrm{f,f}} - \mathbf{K}_{\mathrm{f,f}}(\mathbf{K}_{\mathrm{f,f}}+\sigma^2\mathbf{I})^{-1}\,\mathbf{K}_{\mathrm{f,f}}). \tag{2.9}$$

Since the conditional posterior of $\mathbf{f}$ is Gaussian, the posterior process, or distribution $p(\tilde{f}|\mathcal{D})$, is also Gaussian. By placing the mean and covariance from (2.9) in the equations (2.5) and (2.7) we obtain the predictive distribution

$$\tilde{f}|\mathcal{D},\theta,\sigma^2 \sim \mathcal{GP}\left(m_{\mathrm{p}}(\tilde{\mathbf{x}}), k_{\mathrm{p}}(\tilde{\mathbf{x}},\tilde{\mathbf{x}}')\right), \tag{2.10}$$

where the mean $m_{\mathrm{p}}(\tilde{\mathbf{x}}) = k(\tilde{\mathbf{x}},\mathbf{X})(\mathbf{K}_{\mathrm{f,f}}+\sigma^2\mathbf{I})^{-1}\,\mathbf{y}$ and covariance $k_{\mathrm{p}}(\tilde{\mathbf{x}},\tilde{\mathbf{x}}') = k(\tilde{\mathbf{x}},\tilde{\mathbf{x}}') - k(\tilde{\mathbf{x}},\mathbf{X})(\mathbf{K}_{\mathrm{f,f}}+\sigma^2\mathbf{I})^{-1}k(\mathbf{X},\tilde{\mathbf{x}}')$. The predictive distribution for new observations $\tilde{\mathbf{y}}$ can be obtained by integrating $p(\tilde{\mathbf{y}}|\mathcal{D},\theta,\sigma^2) = \int p(\tilde{\mathbf{y}}|\tilde{\mathbf{f}},\sigma^2)p(\tilde{\mathbf{f}}|\mathcal{D},\theta,\sigma^2)d\tilde{\mathbf{f}}$. The result is, again, Gaussian with mean $\mathrm{E}_{\tilde{\mathbf{f}}|\mathcal{D},\theta}[\tilde{\mathbf{f}}]$ and covariance $\mathrm{Cov}_{\tilde{\mathbf{f}}|\mathcal{D},\theta}[\tilde{\mathbf{f}}] + \sigma^2\mathbf{I}$.

2.1.3 Laplace approximation

In the Laplace approximation, the mean is approximated by the posterior mode of $\mathbf{f}$ and the covariance by the curvature of the log posterior at the mode. The approximation is constructed from the second order Taylor expansion of $\log p(\mathbf{f}\,|\,\mathbf{y},\theta,\phi)$ around the mode $\hat{\mathbf{f}}$, which gives a Gaussian approximation to the conditional posterior

$$p(\mathbf{f}\,|\mathcal{D},\theta,\phi) \approx q(\mathbf{f}\,|\mathcal{D},\theta,\phi) = \mathrm{N}(\mathbf{f}\,|\hat{\mathbf{f}},\boldsymbol{\Sigma}), \tag{2.11}$$

where $\hat{\mathbf{f}} = \arg\max_{\mathbf{f}} p(\mathbf{f}\,|\mathcal{D},\theta,\phi)$ and $\boldsymbol{\Sigma}^{-1}$ is the Hessian of the negative log conditional posterior at the mode (Gelman et al., 2004; Rasmussen and Williams, 2006):

$$\boldsymbol{\Sigma}^{-1} = -\nabla\nabla\log p(\mathbf{f}\,|\mathcal{D},\theta,\phi)|_{\mathbf{f}=\hat{\mathbf{f}}} = \mathbf{K}_{\mathrm{f,f}}^{-1}+\mathbf{W}, \tag{2.12}$$

where $\mathbf{W}$ is a diagonal matrix with entries $\mathbf{W}_{ii} = \nabla_{f_i}\nabla_{f_i}\log p(y|f_i,\phi)|_{f_i=\hat{f}_i}$. Note that actually the Taylor expansion is made for the unnormalized posterior and the normalization term is computed from the Gaussian approximation.

Here, the approximation scheme is called the Laplace method following Williams and Barber (1998), but essentially the same approximation is named Gaussian approximation by Rue et al. (2009) in their Integrated nested Laplace approximation (INLA) scheme for Gaussian Markov random field models.

The posterior mean of $f(\tilde{\mathbf{x}})$ can be approximated from the equation (2.5) by replacing the posterior mean $\mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta}[\mathbf{f}]$ by $\hat{\mathbf{f}}$. The posterior covariance is approximated similarly by using $(\mathbf{K}_{\mathrm{f,f}}^{-1}+\mathbf{W})^{-1}$ in the place of $\mathrm{Cov}_{\mathbf{f}\,|\mathcal{D},\theta}[\mathbf{f}]$. After some rearrangements and using $\mathbf{K}_{\mathrm{f,f}}^{-1}\,\hat{\mathbf{f}} = \nabla\log p(\mathbf{y}\,|\,\mathbf{f})|_{\mathbf{f}=\hat{\mathbf{f}}}$, the approximate posterior predictive distribution is

$$\tilde{f}|\mathcal{D},\theta,\sigma^2 \sim \mathcal{GP}\left(m_{\mathrm{p}}(\tilde{\mathbf{x}}), k_{\mathrm{p}}(\tilde{\mathbf{x}},\tilde{\mathbf{x}}')\right), \tag{2.13}$$

where the mean and covariance are $m_{\mathrm{p}}(\tilde{\mathbf{x}}) = k(\tilde{\mathbf{x}},\mathbf{X})\nabla\log p(\mathbf{y}\,|\,\mathbf{f})|_{\mathbf{f}=\hat{\mathbf{f}}}$ and $k_{\mathrm{p}}(\tilde{\mathbf{x}},\tilde{\mathbf{x}}') = k(\tilde{\mathbf{x}},\tilde{\mathbf{x}}') - k(\tilde{\mathbf{x}},\mathbf{X})(\mathbf{K}_{\mathrm{f,f}}+\mathbf{W})^{-1}k(\mathbf{X},\tilde{\mathbf{x}}')$. The approximate conditional predictive

densities of new observations $\tilde{y}_i$ can now be evaluated, for example, with quadrature integration over each $\tilde{f}_i$ separately

$$p(\tilde{y}_i|\mathcal{D},\theta,\phi) \approx \int p(\tilde{y}_i|\tilde{f}_i,\phi)q(\tilde{f}_i|\mathcal{D},\theta,\phi)d\tilde{f}_i. \tag{2.14}$$

2.1.4 Expectation propagation algorithm

The Laplace method constructs a Gaussian approximation at the posterior mode and approximates the posterior covariance via the curvature of the log density at that point. The expectation propagation (EP) algorithm (Minka, 2001), for its part, tries to minimize the Kullback-Leibler divergence from the true posterior to its approximation

$$q(\mathbf{f}\,|\mathcal{D},\theta,\phi) = \frac{1}{Z_{\text{EP}}} p(\mathbf{f}\,|\theta) \prod_{i=1}^{n} t_i(f_i|\tilde{Z}_i,\tilde{\mu}_i,\tilde{\sigma}_i^2), \tag{2.15}$$

where the likelihood terms have been replaced by site functions $t_i(f_i|\tilde{Z}_i,\tilde{\mu}_i,\tilde{\sigma}_i^2) = \tilde{Z}_i\,\mathrm{N}(f_i|\tilde{\mu}_i,\tilde{\sigma}_i^2)$ and the normalizing constant by Z_{EP}. EP algorithm updates the site parameters $\tilde{Z}_i$, $\tilde{\mu}_i$ and $\tilde{\sigma}_i^2$ sequentially. At each iteration, first the i'th site is removed from the i'th marginal posterior to obtain a cavity distribution $q_{-i}(f_i) = q(f_i|\mathcal{D},\theta)/t_i(f_i)$. Second step is to find a Gaussian $\hat{q}(f_i)$ to which the Kullback-Leibler divergence from the cavity distribution multiplied by the exact likelihood for that site is minimized, that is $\hat{q}(f_i) = \arg\min_q \mathrm{KL}\left(q_{-i}(f_i)p(y_i|f_i)||q(f_i)\right)$. This is equivalent to matching the first and second moment of the two distributions (Seeger, 2008). The site terms $\tilde{Z}_i$ are scaling parameters which ensure that $Z_{\text{EP}} = p(\mathcal{D}|\theta,\phi)$. After the moments are solved, the parameters of the local approximation t_i are updated so that the new marginal posterior $q_{-i}(f_i)t_i(f_i)$ matches with the moments of $\hat{q}(f_i)$. For last, the parameters of the approximate posterior (2.15) are updated to give

$$q(\mathbf{f}\,|\mathcal{D},\theta,\phi) = \mathrm{N}(\mathbf{f}\,|\,\mathbf{K}_{\mathrm{f,f}}(\mathbf{K}_{\mathrm{f,f}}+\tilde{\Sigma})^{-1}\tilde{\mu}, \mathbf{K}_{\mathrm{f,f}} - \mathbf{K}_{\mathrm{f,f}}(\mathbf{K}_{\mathrm{f,f}}+\tilde{\Sigma})^{-1}\,\mathbf{K}_{\mathrm{f,f}}), \tag{2.16}$$

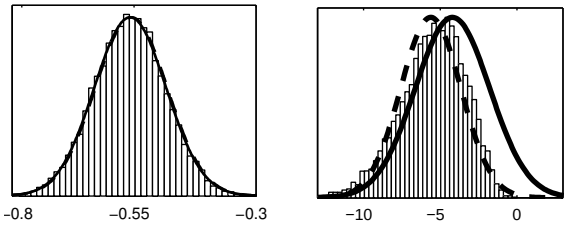
where $\tilde{\Sigma} = \mathrm{diag}[\tilde{\sigma}_1^2,...,\tilde{\sigma}_n^2]$ and $\tilde{\mu} = [\tilde{\mu}_1,...,\tilde{\mu}_n]^{\mathrm{T}}$. The iterations are continued until the convergence. The predictive mean and covariance are again obtained from equations (2.5) and (2.7) analogically to the Laplace approximation.

From the equations (2.9), (2.11), and (2.16) it can be seen that the Laplace and EP approximations are similar to the exact solution with the Gaussian observation model. The diagonal matrices $\mathbf{W}^{-1}$ and $\tilde{\mathbf{\Sigma}}$ correspond to the noise variance $\sigma^2\mathbf{I}$ and, thus, the two approximations can be seen as Gaussian approximations for the likelihood (Nickisch and Rasmussen, 2008).

2.1.5 Markov chain Monte Carlo

The accuracy of the approximations considered this far is limited by the Gaussian form of the approximating function. An approach, which gives exact solution in the limit of an infinite computational time, is the Monte Carlo integration (Robert and Casella, 2004). This is based on sampling from the unnormalized posterior $\frac{1}{Z}p(\mathbf{f}\,|\mathcal{D},\theta,?\phi) \propto p(\mathbf{y}\,|\,\mathbf{f},\phi)p(\mathbf{f}\,|,\mathbf{X},\theta)$ and using the samples to represent the posterior distribution. The posterior marginals can be visualized with histograms and posterior statistics approximated with sample means. For example, the posterior expectation of $\mathbf{f}$ using M samples is

$$\mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta,\phi}[\mathbf{f}] \approx \frac{1}{M}\sum_{i=1}^{M}\mathbf{f}^{(i)}, \tag{2.17}$$



(a) Disease mapping (b) Classification

Figure 2.1: Illustration of the Laplace approximation (solid line), EP (dashed line) and MCMC (histogram) for the conditional posterior of a latent variable $p(f_i|\mathcal{D},\theta)$ in two applications. On the left, a disease mapping problem with Poisson observation model (used in Vanhatalo et al., 2010) where the Gaussian approximation works well. On the right, a classification problem with probit likelihood (used in Vanhatalo and Vehtari, 2010) where the posterior is skewed and the Gaussian approximation is quite poor.

where $\mathbf{f}^{(i)}$ is the i'th sample from the conditional posterior. The problem with Monte Carlo methods is how to draw samples from arbitrary distributions. The challenge can be overcome with Markov chain Monte Carlo methods (Gilks et al., 1996), where one constructs a Markov chain whose stationary distribution is the posterior distribution $p(\mathbf{f}\,|\mathcal{D},\theta,?\phi)$ and uses the Markov chain samples to obtain Monte Carlo estimates. A rather efficient sampling algorithm is hybrid Monte Carlo (HMC) (Duane et al., 1987; Neal, 1996), which utilizes the gradient information of the posterior distribution to direct the sampling in interesting regions. Significant improvement in mixing of the sample chain of the latent variables can be obtained by using the variable transformation discussed in (Christensen et al., 2006; Vanhatalo and Vehtari, 2007). After having the posterior sample of latent variables, we can sample from the posterior predictive distribution of any set of $\tilde{\mathbf{f}}$ simply by sampling with each $\mathbf{f}^{(i)}$ one $\tilde{\mathbf{f}}^{(i)}$ from $p(\tilde{\mathbf{f}}|\,\mathbf{f}^{(i)},\theta,\phi)$, which is given in the equation (1.9). Similarly, we can obtain a sample of $\tilde{\mathbf{y}}$ by drawing one $\tilde{\mathbf{y}}^{(i)}$ for each $\tilde{\mathbf{f}}^{(i)}$ from $p(\tilde{\mathbf{y}}|\tilde{\mathbf{f}}^{(i)},\theta,\phi)$.

2.2 Marginal likelihood given parameters

In Bayesian inference we integrate over the unknown latent values to get the marginal likelihood. In Gaussian case this is straightforward since it has an analytic solution (see equation (2.8)) and log marginal likelihood is

$$\log p(\mathcal{D}|\theta,\sigma) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\,\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I}| - \frac{1}{2}\,\mathbf{y}^{\mathrm{T}}(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}\,\mathbf{y}, \quad (2.18)$$

If the observation model is not Gaussian the marginal likelihood needs to be approximated. The Laplace approximation to the marginal likelihood (denominator in the equation (2.1)) is constructed, for example, by writing

$$p(\mathcal{D}|\theta,\phi) = \int p(\mathbf{y}|\,\mathbf{f},\phi)p(\mathbf{f}\,|\mathbf{X},\theta)d\,\mathbf{f} = \int \exp(g(\mathbf{f}))d\,\mathbf{f}, \quad (2.19)$$

after which a second order Taylor expansion around $\hat{\mathbf{f}}$ is done for $g(\mathbf{f})$. This gives a

Gaussian integral over $\mathbf{f}$ multiplied by a constant, and results in the approximation

$$\log p(\mathcal{D}|\theta,\phi) \approx \log q(\mathcal{D}|\theta,\phi) \propto -\frac{1}{2}\hat{\mathbf{f}}^{\mathrm{T}}\,\mathbf{K}_{\mathrm{f,f}}^{-1}\,\hat{\mathbf{f}} + \log p(\mathbf{y}|\hat{\mathbf{f}},\phi) - \frac{1}{2}\log|\mathbf{B}|, \quad (2.20)$$

where $|\mathbf{B}| = |I + \mathbf{W}^{1/2}\,\mathbf{K}_{\mathrm{f,f}}\,\mathbf{W}^{1/2}|$. Rue et al. (2009) use the same approximation when they define $p(\theta|\mathcal{D}) \approx p(\mathbf{y},\mathbf{f},\theta,\phi)/q(\mathbf{f}\,|\mathcal{D},\theta,\phi)|_{\mathbf{f}=\hat{\mathbf{f}}}$, where the denominator is the Laplace approximation in equation (2.11) (see also Tierney and Kadane, 1986).

EP's marginal likelihood approximation is the normalization constant

$$Z_{\mathrm{EP}} = \int p(\mathbf{f}\,|\mathbf{X},\theta)\prod_{i=1}^{n}\tilde{Z}_i\,\mathrm{N}(f_i|\tilde{\mu}_i,\tilde{\sigma}_i^2)df_i \quad (2.21)$$

in equation (2.15). This is a Gaussian integral multiplied by a constant $\prod_{i=1}^{n}\tilde{Z}_i$, giving

$$\log Z_{\mathrm{EP}} = -\frac{1}{2}\log|K+\tilde{\Sigma}| - \frac{1}{2}\tilde{\mu}^{\mathrm{T}}\left(K+\tilde{\Sigma}\right)^{-1}\tilde{\mu} + C_{\mathrm{EP}}, \quad (2.22)$$

where C_{EP} collects the terms that are not explicit functions of θ or ϕ (there is an implicit dependence through the iterative algorithm, though).

Note that in actual implementation the Laplace and EP marginal likelihood approximations are obtained as byproduct when computing the conditional posterior of $\mathbf{f}$.

2.3 Marginalization over parameters

The previous section treated methods to evaluate exactly (the Gaussian case) or approximately (Laplace and EP approximations) the log marginal likelihood given parameters. In this section we describe approaches for estimating parameters or integrating numerically over them.

2.3.1 Maximum a posterior estimate of parameters

In full Bayesian approach we should integrate over all unknowns. Given we have integrated over the latent variables, it often happens that the posterior of the parameters is peaked or predictions are unsensitive to small changes in parameter values. In such case we can approximate the integral over $p(\theta,\phi|\mathcal{D})$ with maximum a posterior (MAP) estimate

$$\{\hat{\theta},\hat{\phi}\} = \arg\max_{\theta,\phi} p(\theta,\phi|\mathcal{D}) = \arg\min_{\theta,\phi}\left[-\log p(\mathcal{D}|\theta,\phi) - \log p(\theta,\phi)\right]. \quad (2.23)$$

In this approximation, the parameter values are given a point mass one at the posterior mode, and the latent function marginal is approximated as $p(\mathbf{f}\,|\mathcal{D}) \approx p(\mathbf{f}\,|\mathcal{D},\hat{\theta},\hat{\phi})$.

The log marginal likelihood, and thus also the log posterior, is differentiable with respect to the parameters, which allows gradient based optimization. The gradients of the Laplace approximated log marginal likelihood (2.20) can be solved analytically, too. In EP the parameters C_{EP}, $\tilde{\Sigma}$ and $\tilde{\mu}$ can be considered constants when differentiating the function with respect to the parameters (Seeger, 2005), for which reason gradient based optimization is possible also with EP.

The advantage of MAP estimate is that it is relatively easy and fast to evaluate. According to our experience good optimization algorithms need usually at maximum tens of optimization steps to find the mode. However, it may underestimate the uncertainty in parameters.

2.3.2 Grid integration

Grid integration is based on weighted sum of points evaluated on grid

$$p(\mathbf{f}\,|\mathcal{D}) \approx \sum_{i=1}^{M} p(\mathbf{f}\,|\mathcal{D}, \vartheta_i) p(\vartheta_i|\mathcal{D}) \Delta_i. \tag{2.24}$$

Here $\vartheta = [\theta^{\mathrm{T}}, \phi^{\mathrm{T}}]^{\mathrm{T}}$ and Δ_i denotes the area weight appointed to an evaluation point ϑ_i.

The first step in exploring $\log p(\vartheta|\mathcal{D})$ is to find its posterior mode as described in the previous section. After this we evaluate the negative Hessian of $\log p(\vartheta|\mathcal{D})$ at the mode, which would be the inverse covariance matrix for ϑ if the density were Gaussian. If $\mathbf{P}$ is the inverse Hessian (the approximate covariance) with eigendecomposition $\mathbf{P} = \mathbf{U}\mathbf{C}\mathbf{U}^{\mathrm{T}}$, then ϑ can be defined as

$$\vartheta(\mathbf{z}) = \hat{\vartheta} + \mathbf{U}\mathbf{C}^{1/2}\mathbf{z}, \tag{2.25}$$

where $\mathbf{z}$ is a standardized variable. If $p(\vartheta|\mathcal{D})$ were a Gaussian density, then $\mathbf{z}$ would be zero mean normal distributed. This re-parametrization corrects for scale and rotation and simplifies the integration (Rue et al., 2009). The exploration of $\log p(\vartheta|\mathcal{D})$ is started from the mode $\hat{\vartheta}$ and continued so that the bulk of the posterior mass is included in the integration. The grid points are set evenly along the directions $\mathbf{z}$, and thus the area weights Δ_i are equal. This is illustrated in Figure 2.2(a) (see also Rue et al., 2009; Vanhatalo et al., 2010).

The grid search is feasible only for a small number of parameters since the number of grid points grows exponentially with the dimension of the parameter space d. For example, the number of the nearest neighbors of the mode increases as $O(3^d)$, which results in 728 grid points already for $d = 6$. If also the second neighbors are included, the number of grid points increases as $O(5^d)$, resulting in 15624 grid points for six parameters.

2.3.3 Monte Carlo integration

Monte Carlo integration works better than the grid integration in large parameter spaces since its error decreases with a rate that is independent of the dimension (Robert and Casella, 2004). There are two options to find a Monte Carlo estimate for the marginal posterior $p(\mathbf{f}\,|\mathcal{D})$. The first option is to sample only the parameters from their marginal posterior $p(\theta|\mathcal{D})$ or from its approximation (see Figure 2.2(b)). In this case, the posterior marginals are approximated with mixture distributions as in the grid integration. The other option is to run a full MCMC for all the parameters in the model. That is, we sample both the parameters and the latent variables and estimate the needed posterior statistics by sample estimates or by histograms (Neal, 1997; Diggle et al., 1998). The full MCMC is performed by alternate sampling from the conditional posteriors $p(\mathbf{f}\,|\mathcal{D}, \vartheta)$ and $p(\vartheta|\mathcal{D}, \mathbf{f})$. Sampling both the parameters and latent variables is usually awfully slow since there is a strong correlation between them (Vanhatalo and Vehtari, 2007; Vanhatalo et al., 2010). Sampling from the (approximate) marginal, $p(\theta|\mathcal{D})$, is a much easier task since the parameter space is smaller.

The parameters can be sampled from their marginal posterior (or its approximation) either with HMC, slice sampling (SLS) (Neal, 2003) or via importance sampling (Geweke, 1989). In importance sampling, we use a Normal or Student-t proposal distribution with mean $\hat{\vartheta}$ and covariance $\mathbf{P}$ approximated with the negative Hessian of the

log posterior and approximate the integral with

$$p(\mathbf{f}\,|\mathcal{D}) \approx \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} q(\mathbf{f}\,|\mathcal{D}, \vartheta_i) w_i, \tag{2.26}$$

where $w_i = q(\vartheta^{(i)})/g(\vartheta^{(i)})$ are the importance weights. Importance sampling is adequate only if the importance weights do not vary substantially. Thus, the goodness of the Monte Carlo integration can be monitored using the importance weights. The worst scenario occurs when the importance weights are small with high probability and with small probability get very large values (that is the tails of q are much wider than those of g). Problems can be detected by monitoring the cumulative normalized weights and the estimate of the effective sample size $M_{\text{eff}} = 1/\sum_{i=1}^{M} \hat{w}_i^2$, where $\hat{w}_i = w_i/\sum w_i$ (Geweke, 1989; Gelman et al., 2004; Vehtari and Lampinen, 2002). In some situations the naive Gaussian or Student-t proposal distribution is not adequate since the posterior distribution $q(\vartheta|\mathcal{D})$ may be non-symmetric or the covariance estimate $\mathbf{P}$ is poor. In these situations, we use the scaled Student-t proposal distribution, proposed by Geweke (1989). In this approach, the scale of the proposal distribution is adjusted along each main direction defined by $\mathbf{P}$ so that the importance weights are maximized.

Although Monte Carlo integration is more efficient than grid integration for high dimensional problems, it also has its downside. For most examples, few hundred independent samples are enough for reasonable posterior summaries (Gelman et al., 2004), which seems achievable. The problem, however, is that we are not able to draw independent samples from the posterior. Even with a careful tuning of Markov chain samplers the autocorrelation is usually so large that the required sample size is in thousands, which makes Monte Carlo integration computationally very demanding compared to, for example, the MAP estimate.

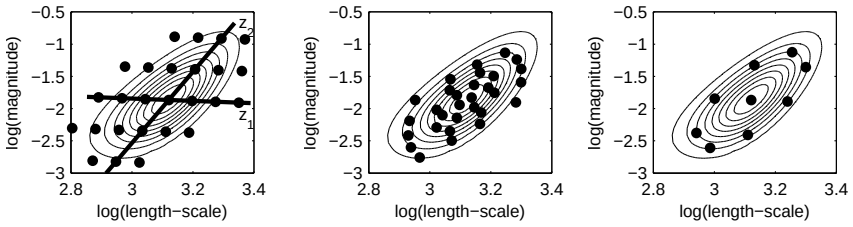
2.3.4 Central composite design

Rue et al. (2009) suggest a central composite design (CCD) for choosing the representative points from the posterior of the parameters when the dimensionality of the parameters, d, is moderate or high. In this setting, the integration is considered as a quadratic design problem in a d dimensional space with the aim at finding points that allow to estimate the curvature of the posterior distribution around the mode. The design used by Rue et al. (2009) and Vanhatalo et al. (2010) is the fractional factorial design (Sanchez and Sanchez, 2005) augmented with a center point and a group of $2d$ star points. In this setting, the design points are all on the surface of a d-dimensional sphere and the star points consist of $2d$ points along each axis, which is illustrated in Figure 2.2(c). The integration is then a finite sum with special weights summarized by the equation (2.24).

The design points are searched after transforming ϑ into $\mathbf{z}$-space, which is assumed to be a standard Gaussian variable. The integration weights can then be determined from the statistics of a standard Gaussian variable $E[\mathbf{z}^{\mathrm{T}}\mathbf{z}] = d$, $E[\mathbf{z}] = \mathbf{0}$ and $E[1] = 1$, which result in

$$\Delta = \left[(n_p - 1) \exp\left(-\frac{df_0^2}{2} \right) \left(f_0^2 - 1 \right) \right]^{-1} \tag{2.27}$$

for the points on the sphere and $\Delta_0 = 1$ for the central point (see appendix Vanhatalo et al., 2010, for a more detailed derivation). The CCD integration speeds up the computations considerably compared to the grid search or Monte Carlo integration since



(a) Grid based. (b) Monte Carlo (c) Central composite design

Figure 2.2: Illustration of the grid based, Monte Carlo and central composite design integration. Contours show the posterior density $q(\log(\vartheta)|\mathcal{D})$ and the integration points are marked with dots. The left figure shows also the vectors $\mathbf{z}$ along which the points are searched in the grid integration and central composite design. The integration is conducted over $q(\log(\vartheta)|\mathcal{D})$ rather than $q(\vartheta|\mathcal{D})$ since the former is closer to Gaussian. Reproduced from (Vanhatalo et al., 2010)

the number of the design points grows very moderately and, for example, for $d = 6$ one needs only 45 points. The accuracy of the CCD is between the MAP estimate and the full integration with the grid search or Monte Carlo. Rue et al. (2009) report good results with this integration scheme, and it has worked well in our experiments as well. CCD tries to incorporate the posterior variance of the parameters in the inference and seems to give good approximation in high dimensions. Since CCD is based on the assumption that the posterior of the parameter is (close to) Gaussian, the densities $p(\vartheta_i|\mathcal{D})$ at the points on the circumference should be monitored in order to detect serious discrepancies from this assumption. These densities are identical if the posterior is Gaussian, and thereby great variability on their values indicates that CCD has failed. The posterior of the parameters may be far from a Gaussian distribution but for a suitable transformation the approximation may work well. For example, the covariance function parameters should be transformed through logarithm as discussed in the section 3.1.

Chapter 3

GPstuff basics: regression and classification

In this chapter, we illustrate the use of GPstuff package with two classical examples, regression and classification. The regression task has Gaussian observation model and forms an important special case of the GP models since it is the only model where we are able to marginalize over the latent variables analytically. Thus, it serves as a good starting point for demonstrating the use of the software package. The classification problem is a usual text book example of tasks with non-Gaussian observation model where we have to utilize the approximate methods discussed in the previous chapter.

This chapter serves also as a general introduction to the GPstuff software package. In the next few sections, we will introduce and discuss many of the functionalities of the package that will be present in more advanced models as well.

3.1 Gaussian process regression: demo_regression1

The demonstration program `demo_regression1` considers a simple regression problem, where we wish to infer a latent function $f(\mathbf{x})$ given measurements corrupted with i.i.d. Gaussian noise as $y_i = f(\mathbf{x}_i) + \epsilon_i$, where $\epsilon_i \sim N(0, \sigma_n^2)$. In this particular example the input vectors $\mathbf{x}_i$ are two-dimensional. This results in the overall model

$$\mathbf{y} \,|\sigma^2 \sim \mathrm{N}(\mathbf{f}, \sigma^2 \mathbf{I}), \tag{3.1}$$

$$f(\mathbf{x})|\theta \sim \mathcal{GP}\left(0, k(\mathbf{x}, \mathbf{x}' \,|\theta)\right), \tag{3.2}$$

$$\theta, \sigma^2 \sim p(\theta)p(\sigma^2). \tag{3.3}$$

We will show how to construct the model with a squared exponential covariance function and how to conduct the inference.

3.1.1 Constructing the model

In GPstuff, the model is constructed in the following three steps:

- create structures that define likelihood and covariance function
- define priors for the parameters

```
gp =
            type: 'FULL'
             lik: [1x1 struct]
              cf: {[1x1 struct]}
    infer_params: 'covariance+likelihood'
    jitterSigma2: 0
```

```
lik =
      type: 'Gaussian'
    sigma2: 0.0400
         p: [1x1 struct]
        fh: [1x1 struct]
```

```
gpcf =
           type: 'gpcf_sexp'
    lengthScale: [1.1000 1.2000]
     magnSigma2: 0.0400
              p: [1x1 struct]
             fh: [1x1 struct]
```

Figure 3.1: The GP, likelihood and covariance function structure in `demo_regression1`.

- create a GP structure, which stores all the above

These three steps are done as follows:

```
lik = lik_gaussian('sigma2', 0.2^2);
gpcf = gpcf_sexp('lengthScale', [1.1 1.2], 'magnSigma2', 0.2^2)

pn = prior_logunif();
lik = lik_gaussian(lik, 'sigma2_prior', pn);

pl = prior_t();
pm = prior_sqrtunif();
gpcf = gpcf_sexp(gpcf, 'lengthScale_prior', pl, 'magnSigma2_prior', pm);

gp = gp_set('type', 'FULL', 'lik', lik, 'cf', {gpcf});
```

Here `lik_gaussian` initializes the likelihood function corresponding to Gaussian observation model (`lik_gaussian` provides code for both the likelihood function and observation model, but for naming simplicity, `lik`-prefix is used) and its parameter values. Function `gpcf_sexp` initializes the covariance function and its parameter values. `lik_gaussian` returns structure `lik` and `gpcf_sexp` returns `gpcf` that contain all the information needed in the evaluations (function handles, parameter values etc.). The next five lines create the prior structures for the parameters of the likelihood and the covariance function, which are set in the likelihood and covariance function structures. The prior for noise variance is usual uninformative log-uniform. The priors for length-scale and magnitude are the Student-t distribution which works as a weakly informative prior (Gelman, 2006) (note that prior is set to the square root of the magnSigma2, i.e., magnitude). The last line creates the GP structure by giving it the likelihood and covariance function and setting the type to `'FULL'` which means that the model is a regular GP without sparse approximations (see section 4 for other types). Type `'FULL'` is the default, and could be left out in this case.

In the GPstuff toolbox, the Gaussian process structure is the fundamental unit, around which all the other blocks of the model are collected. This is illustrated in Figure 3.1. In the `gp` structure `type` defines the type of the model, `likelih` defines the likelihood structure, `cf` contains the covariance function structure, `infer_params` defines which parameters are inferred (covariance, likelihood etc.) and will be discussed in detail in the chapter 4, and `jitterSigma2` is a small constant which is added in the diagonal of the covariance matrix to make it numerically more stable. If

there were more than one covariance function, they would be handled additively. See section 6.2 for details. The likelihood and covariance structure are similar to the GP structure. The likelihood structure contains the type of the likelihood, its parameters (here `sigma2`), prior structure (`p`) and structure holding function handles for internal use (`fh`). The covariance structure contains the type of the covariance, its parameters (here `lengthScale` and `magnSigma2`), prior structure (`p`) and structure holding function handles for internal use (`fh`).

Using the constructed GP structure, we can evaluate basic summaries such as covariance matrices, make predictions with the present parameter values etc. For example, the covariance matrices $\mathbf{K}_{\mathrm{f,f}}$ and $\mathbf{C} = \mathbf{K}_{\mathrm{f,f}} + \sigma^2_{\mathrm{noise}}\mathbf{I}$ for three two-dimensional input vectors can be evaluated as follows:

```
example_x = [-1 -1 ; 0 0 ; 1 1];
[K, C] = gp_trcov(gp, example_x)
K =
    0.0400    0.0054    0.0000
    0.0054    0.0400    0.0054
    0.0000    0.0054    0.0400
C =
    0.0800    0.0054    0.0000
    0.0054    0.0800    0.0054
    0.0000    0.0054    0.0800
```

Here `K` is $\mathbf{K}_{\mathrm{f,f}}$ and `C` is $\mathbf{K}_{\mathrm{f,f}} + \sigma^2_{\mathrm{noise}}\mathbf{I}$.

3.1.2 MAP estimate for the hyperparameters

MAP estimate for the parameters can be obtained with function `gp_optim`, which works as a wrapper for usual gradient based optimization functions. It can be used as follows:

```
opt=optimset('TolFun',1e-3,'TolX',1e-3,'Display','iter');
gp=gp_optim(gp,x,y,'optimf',@fminscg,'opt',opt);
```

`gp_optim` takes a GP structure, training input $\mathbf{x}$, training target $\mathbf{y}$ (which are defined in `demo_regression1`) and options, and returns a GP structure with optimized hyper parameter values. By default `gp_optim` uses `fminscg` function, but `gp_optim` can use also, for example, `fminlbfgs` or `fminunc`. Optimization options are set with `optimset` function. All the estimated parameter values can be easily checked using the function `gp_pak`, which packs all the parameter values from all the covariance function structures in one vector, usually using log-transformation (other transformations are also possible). The second output argument of `gp_pak` lists the labels for the parameters:

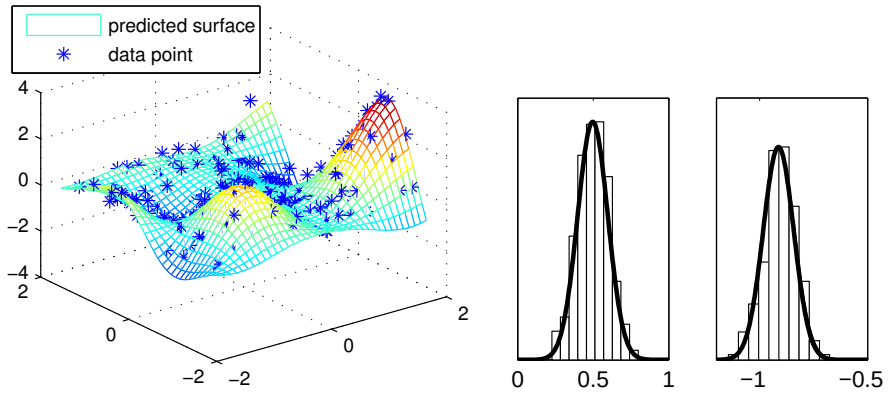
```
[w,s] = gp_pak(gp);
disp(exp(w)), disp(s)
```

It is also possible to set the parameter vector of the model to desired values using `gp_unpak`.

```
gp = gp_unpak(gp,w);
```

It is possible to control which parameters optimized in two ways. 1) If any parameter is given an empty prior it is considered fixed. 2) It is possible to use `gp_set` to set option `infer_params` to control which groups of parameters covariance, likelihood or inducing are meant to be inferred.

To make predictions for new locations $\tilde{\mathbf{x}}$, given the training data $(\mathbf{x}, \mathbf{y})$, we can use the `gp_pred` function, which returns the posterior predictive mean and variance for each $f(\tilde{\mathbf{x}})$ (see equation (2.10)). This is illustrated below where we create a regular grid where the posterior mean and variance are computed. The posterior mean $m_p(\tilde{\mathbf{x}})$ and the training data points are shown in Figure 3.2.



(a) The predictive mean and training data.

(b) The marginal posterior predictive distributions $p(f_i|\mathcal{D})$.

Figure 3.2: The predictive mean surface, training data, and the marginal posterior for two latent variables in `demo_regression1`. Histograms show the MCMC solution and the grid integration solution is drawn with a line.

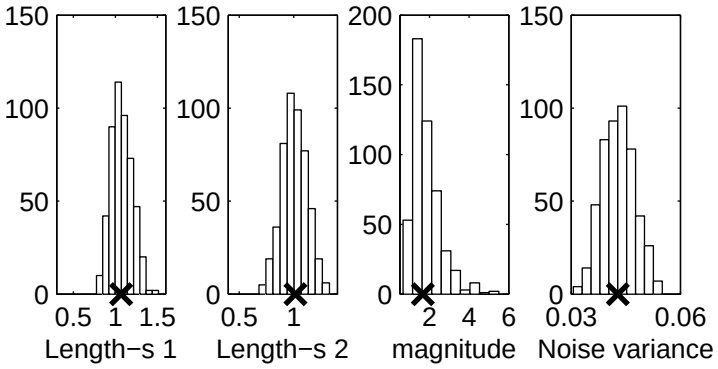


Figure 3.3: The posterior distribution of the hyperparameters together with MAP estimate (black cross). The results are from `demo_regression1`.

```
[xt1,xt2]=meshgrid(-1.8:0.1:1.8,-1.8:0.1:1.8);
xt=[xt1(:) xt2(:)];
[Eft_map, Varft_map] = gp_pred(gp, x, y, xt);
```

3.1.3 Marginalization over parameters with grid integration

To integrate over the parameters we can use any method described in the section 2.3. The grid integration is performed with the following line:

```
[gp_array, P_TH, th, Eft_ia, Varft_ia, fx_ia, x_ia] = ...
                                gp_ia(gp, x, y, xt, 'int_method', 'grid');
```

`gp_ia` returns an array of GPs (`gp_array`) for parameter values `th` ($[\vartheta_i]_{i=1}^M$) with weights `P_TH` ($[p(\vartheta_i|\mathcal{D})\Delta_i]_{i=1}^M$). Since we use the grid method the weights are propor-

tional to the marginal posterior and $\Delta_i \equiv 1 \forall i$ (see section 2.3.2). The last four outputs are returned only if the prediction locations `xt` (representing $\tilde{\mathbf{X}}$) are given. `Ef_ia` and `Varf_ia` contain the predictive mean and variance at the prediction locations. The last two output arguments can be used to plot the predictive distribution $p(\tilde{f}_i|\mathcal{D})$ as demonstrated in Figure 3.2. `x_ia` contains a regular grid of values $\tilde{f}_i$ and `fx_ia` contains $p(\tilde{f}_i|\mathcal{D})$ at those values.

The `gp_ia` optimizes the hyperparameters to their posterior mode, evaluates the Hessian $\mathbf{P}^{-1}$ with finite differences, and makes a grid search starting from the mode. Since we give the prediction inputs `p` (representing $\tilde{\mathbf{X}}$) the integration method returns also the predictive distributions for those locations. Otherwise `gp_ia` would return only the first three arguments, which contain the array of GP's (`gp_array`) for hyperparameter values `th` ($[\vartheta_i]_{i=1}^M$) with weights `P_TH` ($[p(\vartheta_i|\mathcal{D})\Delta_i]_{i=1}^M$). Since we use the grid method the weights are proportional to the marginal posterior and $\Delta_i \equiv 1 \forall i$ (see section 2.3.2). `Ef_ia` and `Varf_ia` contain the predictive mean and variance at the prediction locations `p`. The last two output arguments can be used to plot the predictive distribution $p(\tilde{f}_i|\mathcal{D})$ at input location $\tilde{\mathbf{x}}_i$ (which is the i'th row of `p`) as demonstrated in the Figure 3.2. `x_ia` contains a regular grid of values $\tilde{f}_i$ and `fx_ia` contains $p(\tilde{f}_i|\mathcal{D})$ at those values.

3.1.4 Marginalization over parameters with MCMC

The main function for conducting Markov chain sampling is `gp_mc`, which loops through all the specified samplers in turn and saves the sampled parameters in a record structure. In later sections, we will discuss models where also latent variables are sampled, but now we concentrate on the covariance function parameters. Sampling and predictions can be made as follows:

```
hmc_opt = hmc2_opt;
[rfull,g,opt] = gp_mc(gp, x, y, 'nsamples', 400, 'repeat', 5, 'hmc_opt', hmc_opt);
rfull = thin(rfull, 50, 2);
[Eft_mc, Varft_mc] = gpmc_preds(rfull, x, y, xt);
```

The `gp_mc` function generates `nsamples` Markov chain samples. Between consecutive samples `repeat` (5) sampler iterations are done. At each iteration `gp_mc` runs the actual samplers. For example, giving the option `'hmc_opt'` tells that `gp_mc` should run the hybrid Monte Carlo sampler with sampling options stored in the structure `hmc_opt`. The default sampling options for HMC are set by `hmc2_opt` function. The function `thin` removes the burn-in from the sample chain (here 50) and thins the chain with user defined parameter (here 2). This way we can decrease the autocorrelation between the remaining samples. GPstuff provides also other diagnostic tools for Markov chains and in detailed MCMC analysis we should use several starting points and monitor the convergence carefully (Gelman et al., 2004; Robert and Casella, 2004). Figure 3.3 shows an example of approximating a posterior distribution of parameters with Monte Carlo sampling. The function `gpmc_preds` returns the conditional predictive mean and variance for each sampled parameter value. These are $\mathrm{E}_{p(\mathbf{f}|\mathbf{X},\mathcal{D},\vartheta^{(s)})}[\tilde{\mathbf{f}}], s = 1, ..., M$ and $\mathrm{Var}_{p(\mathbf{f}|\mathbf{X},\mathcal{D},\vartheta^{(s)})}[\tilde{\mathbf{f}}], s = 1, ..., M$, where M is the number of samples. Marginal predictive mean and variance can be computed directly with `gp_pred`.

3.2 Gaussian process classification

We will now consider a binary GP classification (see `demo_classific`) with observations, $y_i \in \{-1, 1\}, i = 1, ..., n$, associated with inputs $\mathbf{X} = \{\mathbf{x}?\}_{i=1}^n$. The

observations are considered to be drawn from a Bernoulli distribution with a success probability $p(y_i = 1|\mathbf{x}_i)$. The probability is related to the latent function via a sigmoid function that transforms it to a unit interval. GPstuff provides a probit and logit transformation, which lead to observation models

$$p_{\text{probit}}(y_i|f(\mathbf{x}_i)) = \Phi(y_i f(\mathbf{x}_i)) = \int_{-\infty}^{y_i f(\mathbf{x}_i)} N(z|0,1)dz \tag{3.4}$$

$$p_{\text{logit}}(y_i|f(\mathbf{x}_i)) = \frac{1}{1+\exp(-y_i f(\mathbf{x}_i))}. \tag{3.5}$$

Since the likelihood is not Gaussian we need to use approximate inference methods, discussed in the section 2.1. We will conduct the inference with Laplace approximation, EP and full MCMC. With the two analytic approximations we optimize the parameters to their MAP estimates and use those point estimates for prediction. In MCMC approach, we sample both the parameters and the latent variables. For a detailed discussion on the differences between MCMC, Laplace and EP approaches in classification setting see (Kuss and Rasmussen, 2005; Nickisch and Rasmussen, 2008).

3.2.1 Constructing the model

The model construction for the classification follows closely the steps presented in the previous section. The model is constructed as follows:

```
lik = lik_probit();
gpcf = gpcf_sexp('lengthScale', [0.9 0.9], 'magnSigma2', 10);
gp = gp_set('lik', lik, 'cf', {gpcf}, 'jitterSigma2', 1e-9);
```

The above lines first initialize the likelihood function, the covariance function and the GP structure. The default uniform priors for length-scale and magnitude are used. The model construction and the inference with logit likelihood would be similar with `lik_logit`. A small jitter value is added to the diagonal of the training covariance to make certain matrix operations more stable.

3.2.2 Inference with Laplace approximation

The MAP estimate for the parameters can be found using `gp_optim` as

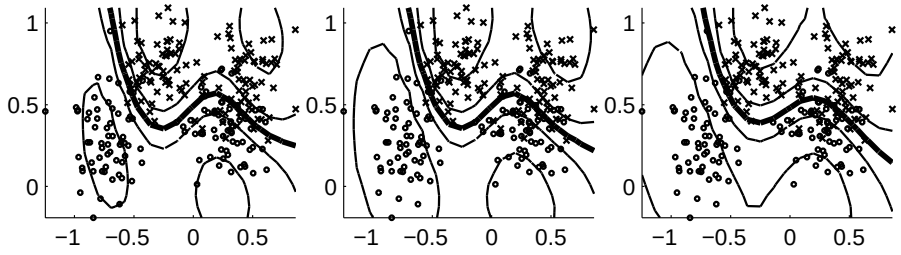
```
gp = gp_set(gp, 'latent_method', 'Laplace');
gp = gp_optim(gp,x,y,'opt',opt);
```

The first line defines which inference method is used for the latent variables. It initializes the Laplace algorithm and sets needed fields in the GP structure. The default method for latent variables is Laplace, so this line could be omitted. The next line uses default `fminscg` optimization function with same the options as above.

`gp_pred` can be used to obtain the mean and variance for the latent variables and new observations together with the predictive probability for a test observation.

```
[Eft_la, Varft_la, Eyt_la, Varyt_la, pyt_la] = ...
                          gp_pred(gp, x, y, xt, 'yt', ones(size(xt,1),1) );
```

The first four input arguments are the same as in the section 3, the fifth and sixth arguments are a parameter-value pair. Parameter `yt` tells that we give test observations related to `xt` as optional inputs for `gp_pred`. Here we want to evaluate the probability to observe class 1 and thus we give a vector of ones as test observations. The probability densities for test observations are returned in the fifth output argument `pyt`. The training data together with predicted class probability contours is visualized in Figure 3.4(a).



(a) Laplace approximation. (b) Expectation propagation. (c) Markov chain Monte Carlo.

Figure 3.4: The class probability contours for Laplace approximation, EP, and MCMC solution. The strongest line in the middle is the 50% probability (the decision boundary) and the thinner lines are 2.5%, 25%, 75% and 97.5% probability contours. It is seen that the decision boundary is approximated similarly with all the methods but there is a larger difference elsewhere.

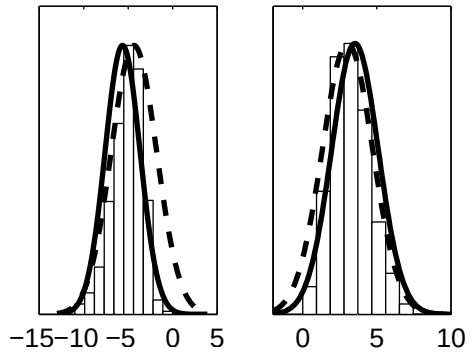


Figure 3.5: The marginal posterior for two latent variables in `demo_classific1`. Histogram is the MCMC solution, dashed line is the Laplace approximation and full line the EP estimate. The left figure is for latent variable at location $[-1.0, 0.2]$ and the right figure at $[0.9, 0.9]$.

3.2.3 Inference with expectation propagation

The EP approximation is used similarly as the Laplace approximation. We only need to set the latent method to EP:

```
gp = gp_set(gp, 'latent_method', 'EP');
gp = gp_optim(gp,x,y,'opt',opt);
[Eft_ep, Varft_ep, Eyt_ep, Varyt_ep, pyt_ep] = ...
                    gp_pred(gp, x, y, xt, 'yt', ones(size(xt,1),1) );
```

The EP solution for the predicted class probabilities is visualized in the Figure 3.4(b), and the Figure 3.5 shows the marginal predictive posterior for two latent variables.

3.2.4 Inference with MCMC

The MCMC solution with non-Gaussian likelihood is found similarly as with the Gaussian model discussed earlier. The major difference is that now we need to sample also the latent variables. We need to set the latent method to MCMC:

```
gp = gp_set(gp, 'latent_method', 'MCMC', 'jitterSigma2', 1e-4);
```

For MCMC we need to add a larger jitter value to the diagonal to avoid numerical problems. By default sampling from the latent value distribution $p(\mathbf{f}\,|\theta, \mathcal{D})$ is done using `@scaled_mh`, which implements scaled Metropolis Hastings algorithm (Neal, 1998). Other sampler provided by the GPstuff is a scaled HMC `scaled_hmc` (Vanhatalo and Vehtari, 2007). Below, the actual sampling is performed similarly as with the Gaussian likelihood.

```
% Set the parameters for MCMC...
hmc_opt=hmc2_opt;
hmc_opt.steps=10;
hmc_opt.stepadj=0.05;
hmc_opt.nsamples=1;
latent_opt.display=0;
latent_opt.repeat = 20;
latent_opt.sample_latent_scale = 0.5;
hmc2('state', sum(100*clock))

[r,g,opt]=gp_mc(gp, x, y, 'hmc_opt', hmc_opt, 'latent_opt', latent_opt, 'nsamples', 1, 'repeat', 15);

% re-set some of the sampling options
hmc_opt.steps=4;
hmc_opt.stepadj=0.05;
latent_opt.repeat = 5;
hmc2('state', sum(100*clock));

% Sample
[rgp,g,opt]=gp_mc(gp, x, y, 'nsamples', 400, 'hmc_opt', hmc_opt,
             'latent_opt', latent_opt, 'record', r);

% Make predictions
[Efs_mc, Varfs_mc, Eys_mc, Varys_mc, Pys_mc] = ...
           gpmc_preds(rgp, x, y, xt, 'yt', ones(size(xt,1),1));
Pyt_mc = mean(Pys_mc,2);
```

The HMC options are set into the `hmc_opt` structure in a similar manner as in the regression example. Since we are sampling also the latent variables we need to give options for their sampler as well. These options are set into the `latent_opt` structure. The options specific to `gp_mc` are given with parameter-value pairs `'nsamples', 1, 'repeat', 15`. The above lines demonstrate also how the sampling can be continued from an old sample chain. The first call for `gp_mc` returns a record structure with only one sample. This record is then given as an optional parameter for `gp_mc` in the second call for it. The sampling is continued from the previously sampled parameter values. The sampling options are also modified between the two successive sampling phases. The modified options are then given to `gp_mc`. The line

`hmc2('state', sum(100*clock));` re-sets the state of the random number generators in the HMC sampler. The last two lines evaluate the predictive statistics similarly to the EP and Laplace approximations. However, now the statistics are matrices whose columns contain the result for one MCMC sample each. The `gp_mc` function handles the sampling so that it first samples the latent variables from $p(\mathbf{f}|\theta, \mathcal{D})$ using the scaled Metropolis Hastings after which it samples the hyperparameters from $p(\theta|\,\mathbf{f}, \mathcal{D})$. This is repeated until `nsamples` samples are drawn.

The MCMC solution for the predicted class probabilities is visualized in Figure 3.4(c), and Figure 3.5 shows the marginal predictive posterior for two latent variables. It can be seen that there are little differences between the three different approximations. Here MCMC is the most accurate, then comes EP and Laplace is the worst. However, the inference times line up in the opposite order. The difference between the approximations is not always this large. For example, in spatial epidemiology with Poisson observation model (see Section 7.2), Laplace and EP approximations work, in our experience, practically as well as MCMC.

Chapter 4

Sparse Gaussian processes

The challenges with using Gaussian process models are the fast increasing computational time and memory requirements. The evaluation of the inverse and determinant of the covariance matrix in the log marginal likelihood (or its approximation) and its gradient scale as $O(n^3)$ in time, which restricts the implementation of GP models to moderate size data sets. For this reason there are number of sparse Gaussian processes introduced in the literature.

4.1 Compactly supported covariance functions

A compactly supported covariance function gives zero correlation between data points whose distance exceeds a certain threshold leading to a sparse covariance matrix. The challenge in constructing CS covariance functions is to guarantee their positive definiteness. A globally supported covariance function cannot be cut arbitrarily to obtain a compact support, since the resulting function would not, in general, be positive definite. Sansò and Schuh (1987) provide one of the early implementations of spatial prediction with CS covariance functions. Their functions are build by selfconvoluting finite support symmetric kernels (such as a linear spline). These are, however, special functions for one or two dimensions. Wu (1995) introduced radial basis functions with a compact support and Wendland (1995) developed them further. Later, for example, Gaspari and Cohn (1999), Gneiting (1999, 2002), and Buhmann (2001) have worked more on the subject. The CS functions implemented in GPstuff are Wendland's piecewise polynomials $k_{\text{pp},q}$ (Wendland, 2005), such as

$$k_{\text{pp},2} = \frac{\sigma^2_{\text{pp}}}{3}(1-r)_+^{j+2}\left((j^2+4j+3)r^2+(3j+6)r+3\right), \tag{4.1}$$

where $j = \lfloor d/2 \rfloor + 3$ and $r^2 = \sum_{k=1}^{d}(x_{i,k}-x_{j,k})^2/l_k^2$. These functions correspond to processes that are q times mean square differentiable and are positive definite up to an input dimension d. Thus, the degree of the polynomial has to be increased alongside the input dimension. The dependence of CS covariance functions to the input dimension is very fundamental. There are no radial compactly supported functions that are positive definite on every $\Re^d$ but they are always restricted to a finite number of dimensions (see e.g. Wendland, 1995, theorem 9.2).

The key idea of using CS covariance functions is that, roughly speaking, one uses only the nonzero elements of the covariance matrix in the calculations. This may speed

up the calculations substantially since in some situations only a fraction of the elements of the covariance matrix are non-zero. In practice, efficient sparse matrix routines are needed (Davis, 2006). These are nowadays a standard utility in many statistical computing packages, such as MATLAB or R, or available as an additional package for them. The CS covariance functions have been rather widely studied in the geostatistics applications. The early works concentrated on their theoretical properties and aimed to approximate the known globally supported covariance functions (Gneiting, 2002; Furrer et al., 2006; Moreaux, 2008). There the computational speed-up is obtained using efficient linear solvers for the prediction equation $\tilde{\mathbf{f}} = \mathbf{K}_{\tilde{\mathrm{f}},\mathrm{f}}(\mathbf{K}_{\mathrm{f,f}} + \sigma^2)^{-1}\,\mathbf{y}$. The parameters are fitted to either the empirical covariance or a globally supported covariance function. Kaufman et al. (2008) study the maximum likelihood estimates for tapered covariance functions (i.e. products of globally supported and CS covariance functions) where the magnitude can be solved analytically and the length-scale is optimized using a line search in one dimension. The benefits from a sparse covariance matrix have been immediate since the problems collapse to solving sparse linear systems. However, utilizing the gradient of the log posterior of the parameters needs some extra sparse matrix tools.

The problematic part is the trace in the derivative of the log marginal likelihood, for example,

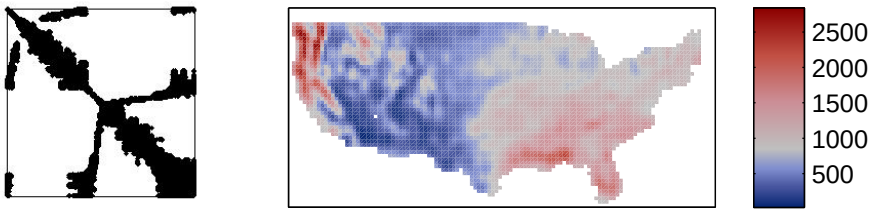
$$\begin{aligned}\frac{\partial}{\partial\theta}\log p(\mathbf{y}|\mathbf{X},\theta,\sigma^2) =& \frac{1}{2}\mathbf{y}^{\mathrm{T}}(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}\frac{\partial(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})}{\partial\theta}(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}\mathbf{y} \\ & - \frac{1}{2}\mathrm{tr}\left((\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}\frac{\partial(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})}{\partial\theta}\right). \qquad (4.2)\end{aligned}$$

The trace would require the full inverse of the covariance matrix if evaluated naively. Luckily, Takahashi et al. (1973) introduced an algorithm whereby we can evaluate a sparsified version of the inverse of a sparse matrix. This can be utilized in the gradient evaluations as described by Vanhatalo and Vehtari (2008). The same problem was considered by Storkey (1999) who used the covariance matrices of Toeplitz form, which are fast to handle due their banded structure. However, constructing Toeplitz covariance matrices is not possible in two or higher dimensions without approximations. EP algorithm requires also special considerations with CS covariance functions. The posterior covariance in EP (2.16) does not remain sparse, and thereby it has to be expressed implicitly during the updates. This issue is discussed in (Vanhatalo et al., 2010; Vanhatalo and Vehtari, 2010).

Compactly supported covariance functions in GPstuff

The demo `demo_regression_ppcs` contains a regression example with fairly large data that contain US annual precipitation summaries from year 1995 for 5776 observation stations. The data was previously used by Vanhatalo and Vehtari (2008) and can be downloaded from (http://www.image.ucar.edu/Data/). The model is the same as described in the section 3.1 but now we use the piecewise polynomial covariance function $k_{\mathrm{pp},2}$ (`gpcf_ppcs2`). GPstuff utilizes the sparse matrix routines from SuiteSparse written by Tim Davis (http://www.cise.ufl.edu/research/sparse/SuiteSparse/) and this package should be installed before using the compactly supported covariance functions.

The user interface of GPstuff makes no difference between globally and compactly supported covariance functions but the code is optimized so that it uses sparse matrix routines whenever the covariance matrix is sparse. Thus, we can construct the model,



(a) The nonzero elements of $\mathbf{K}_{\mathrm{f,f}}$.

(b) The posterior predictive mean surface.

Figure 4.1: The nonzero elements of $\mathbf{K}_{\mathrm{f,f}}$ with $k_{\mathrm{pp},2}$ function, and the posterior predictive mean of the latent function in the US precipitation data set.

find the MAP estimate for the parameters and predict to new input locations in a familiar way:

```
pn = prior_t('nu', 4, 's2', 0.3);
lik = lik_gaussian('sigma2', 1, 'sigma2_prior', pn);
pl2 = prior_gamma('sh', 5, 'is', 1);
pm2 = prior_sqrtt('nu', 1, 's2', 150);
gpcf2 = gpcf_ppcs2('nin', nin, 'lengthScale', [1 2], 'magnSigma2', 3, ...
                   'lengthScale_prior', pl2, 'magnSigma2_prior', pm2);
gp = gp_set('lik', lik, 'cf', {gpcf}, 'jitterSigma2', 1e-8);
gp = gp_optim(gp,x,y,'opt',opt);
Eft = gp_pred(gp, x, y, xt);
```

With this data the covariance matrix is rather sparse since only about 5% of its elements are non-zero. The following lines show how to evaluate the sparsity of the covariance matrix and how to plot the non-zero structure of the matrix. The structure of the covariance matrix is plotted after the AMD permutation (Davis, 2006) in Figure 4.1.

```
K = gp_trcov(gp,x);
nnz(K) / prod(size(K))

p = amd(K);
spy(K(p,p), 'k')
```

In the section 7.2.2 we discuss a demo with non-Gaussian likelihood and compactly supported covariance function.

4.2 FIC and PIC sparse approximations

Snelson and Ghahramani (2006) proposed a sparse pseudo-input Gaussian process (SPGP), which Quiñonero-Candela and Rasmussen (2005) named later fully independent conditional (FIC). The original idea in SPGP was that the sparse approximation is used only in the training phase and predictions are conducted using the exact covariance matrix, where the word 'training' comes to the name. If the approximation is used also for the predictions, the word training should drop out leading to FIC. In this case, FIC can be seen as a non-stationary covariance function on its own (Snelson, 2007). The partially independent conditional (PIC) sparse approximation is an extension of FIC (Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2007), and they are both treated here following Quiñonero-Candela and Rasmussen

(2005). The approximations are based on introducing an additional set of latent variables $\mathbf{u} = \{u_i\}_{i=1}^m$, called *inducing variables*. These correspond to a set of input locations $\mathbf{X}_u$, called *inducing inputs*. The latent function prior is approximated as

$$\mathbf{X}, \mathbf{X}_u, \mathbf{u}, \theta) p(\mathbf{u} \,|\mathbf{X}_u, \theta) \mathrm{d}\mathbf{u}, \tag{4.3}$$

where $q(\mathbf{f} \,|\mathbf{X}, \mathbf{X}_u, \mathbf{u}, \theta)$ is an inducing conditional. The above decomposition leads to the exact prior if the true conditional $\mathbf{f} \,|\mathbf{X}, \mathbf{X}_u, \mathbf{u}, \theta \sim \mathrm{N}(\mathbf{K}_{\mathrm{f,u}} \mathbf{K}_{\mathrm{u,u}}^{-1} \mathbf{u}, \mathbf{K}_{\mathrm{f,f}} - \mathbf{K}_{\mathrm{f,u}} \mathbf{K}_{\mathrm{u,u}}^{-1} \mathbf{K}_{\mathrm{u,f}})$ is used. However, in FIC framework the latent variables are assumed to be conditionally independent given $\mathbf{u}$, in which case the inducing conditional factorizes $q(\mathbf{f} \,|\mathbf{X}, \mathbf{X}_u, \mathbf{u}, \theta) = \prod q_i(f_i|\mathbf{X}, \mathbf{X}_u, \mathbf{u}, \theta)$. In PIC latent variables are set in blocks which are conditionally independent of each others, given $\mathbf{u}$, but the latent variables within a block have a multivariate normal distribution with the original covariance. The approximate conditionals of FIC and PIC can be summarized as

$$q(\mathbf{f} \,|\mathbf{X}, \mathbf{X}_u, \mathbf{u}, \theta, \mathbf{M}) = \mathrm{N}(\mathbf{f} \,|\, \mathbf{K}_{\mathrm{f,u}} \mathbf{K}_{\mathrm{u,u}}^{-1} \mathbf{u}, \mathrm{mask}\left(\mathbf{K}_{\mathrm{f,f}} - \mathbf{K}_{\mathrm{f,u}} \mathbf{K}_{\mathrm{u,u}}^{-1} \mathbf{K}_{\mathrm{u,f}} | \mathbf{M}\right)), \tag{4.4}$$

where the function $\mathbf{\Lambda} = \mathrm{mask}\,(\cdot|\mathbf{M})$, with matrix $\mathbf{M}$ of ones and zeros, returns a matrix $\mathbf{\Lambda}$ of size $\mathbf{M}$ and elements $\mathbf{\Lambda}_{ij} = [\cdot]_{ij}$ if $\mathbf{M}_{ij} = 1$ and $\mathbf{\Lambda}_{ij} = 0$ otherwise. An approximation with $\mathbf{M} = \mathbf{I}$ corresponds to FIC and an approximation where $\mathbf{M}$ is block diagonal corresponds to PIC. The inducing inputs are given a zero-mean Gaussian prior $\mathbf{u} \,|\theta, \mathbf{X}_u \sim \mathrm{N}(\mathbf{0}, \mathbf{K}_{\mathrm{u,u}})$ so that the approximate prior over latent variables is

$$q(\mathbf{f} \,|\mathbf{X}, \mathbf{X}_u, \theta, \mathbf{M}) = \mathrm{N}(\mathbf{f} \,|\mathbf{0}, \mathbf{K}_{\mathrm{f,u}} \mathbf{K}_{\mathrm{u,u}}^{-1} \mathbf{K}_{\mathrm{u,f}} + \mathbf{\Lambda}), \tag{4.5}$$

The matrix $\mathbf{K}_{\mathrm{f,u}} \mathbf{K}_{\mathrm{u,u}}^{-1} \mathbf{K}_{\mathrm{u,f}}$ is of rank m and $\mathbf{\Lambda}$ is a rank n (block) diagonal matrix. The prior covariance above can be seen as a non-stationary covariance function of its own where the inducing inputs $\mathbf{X}_u$ and the matrix $\mathbf{M}$ are free parameters similar to parameters, which can be optimized alongside θ (Snelson and Ghahramani, 2006; Lawrence, 2007).

The computational savings are obtained by using the Woodbury-Sherman-Morrison lemma (e.g. Harville, 1997) to invert the covariance matrix in (4.5) as

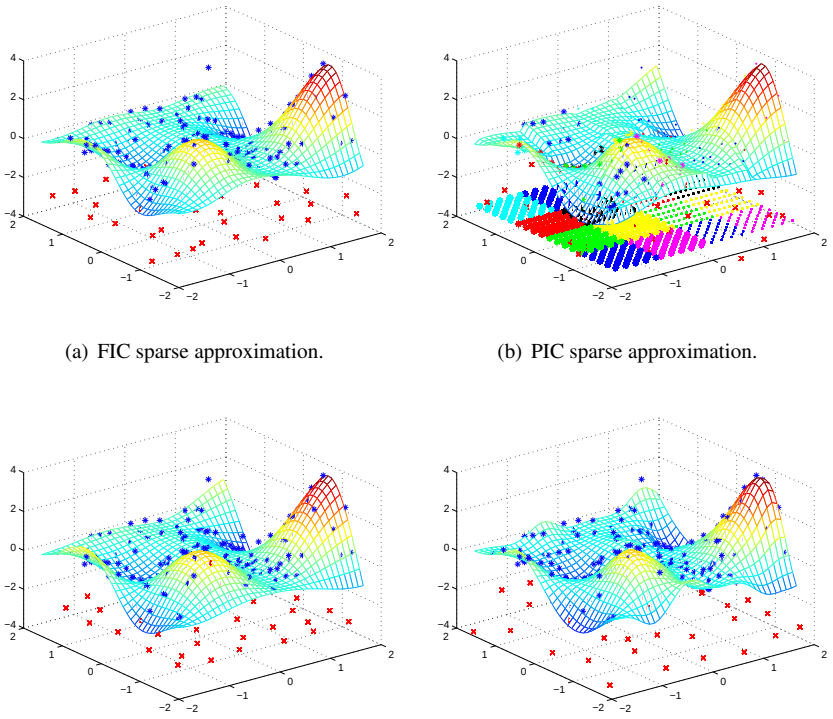
$$(\mathbf{K}_{\mathrm{f,u}} \mathbf{K}_{\mathrm{u,u}}^{-1} \mathbf{K}_{\mathrm{u,f}} + \mathbf{\Lambda})^{-1} = \mathbf{\Lambda}^{-1} - \mathbf{V}\mathbf{V}^{\mathrm{T}}, \tag{4.6}$$

where $\mathbf{V} = \mathbf{\Lambda}^{-1} \mathbf{K}_{\mathrm{f,u}} \mathrm{chol}[(\mathbf{K}_{\mathrm{u,u}} + \mathbf{K}_{\mathrm{u,f}} \mathbf{\Lambda}^{-1} \mathbf{K}_{\mathrm{f,u}})^{-1}]$. There is a similar result also for the determinant. With FIC the computational time is dominated by the matrix multiplications, which need time $O(m^2 n)$. With PIC the cost depends also on the sizes of the blocks in $\mathbf{\Lambda}$. If the blocks were of equal size $b \times b$, the time for inversion of $\mathbf{\Lambda}$ would be $O(n/b \times b^3) = O(nb^2)$. With blocks at most the size of the number of inducing inputs, that is $b = m$, the computational cost in PIC and FIC are similar. PIC approaches FIC in the limit of a block size one and the exact GP in the limit of a block size n (Snelson, 2007).

FIC sparse approximation in GPstuff

The same data that were discussed in the section 3.1 is analyzed with sparse approximations in the demo `demo_regression_sparse1`. The sparse approximation is always a property of the GP structure and we can construct the model similarly to the full GP models:

```
lik = lik_gaussian('sigma2', 0.2^2);
gpcf = gpcf_sexp('lengthScale', [1 1], 'magnSigma2', 0.2^2);
```



(a) FIC sparse approximation.

(b) PIC sparse approximation.

(c) Variational sparse approximation.

(d) DTC/SOR sparse approximation.

Figure 4.2: The posterior predictive mean of the latent function in the `demo_sparseApprox` data set obtained with FIC, PIC, variational and DTC/SOR sparse approximations. The red crosses show the optimized inducing inputs and the block areas for PIC are colored underneath the latent surface.

```
[u1,u2]=meshgrid(linspace(-1.8,1.8,6),linspace(-1.8,1.8,6));
X_u = [u1(:) u2(:)];
gp_fic = gp_set('type', 'FIC', 'lik', lik, 'cf', {gpcf}, ...
                'X_u', X_u, 'jitterSigma2', 1e-4);
```

The difference is that we have to define the type of the sparse approximation, here `'FIC'`, and set the inducing inputs `X_u` in the GP structure. The posterior predictive mean and the inducing inputs are shown in Figure 4.2.

Since the inducing inputs are considered as extra parameters common to all of the covariance functions (there may be more than one covariance function in additive models) they are set in the GP structure instead of the covariance function structure. If we want to optimize the inducing inputs alongside the parameters, we need to use `gp_set` to set option `infer_params` to include `'inducing'`. Sometimes, for example in spatial problems, it is better to fix the inducing inputs (see Vanhatalo et al., 2010) or it may be more efficient to optimize the parameters and inducing inputs separately, so that we iterate the separate optimization steps until convergence.

PIC sparse approximation in GPstuff

In PIC, in addition to defining the inducing inputs, we need to appoint every data point in a block. The block structure is common to all covariance functions, similarly to the inducing inputs, for which reason the block information is stored in the GP structure. With this data set we divide the two dimensional input space into 16 equally sized square blocks and appoint the training data into these according to the input coordinates. This and the initialization of the GP structure are done as follows:

```
% Initialize the inducing inputs in a regular grid over the input space
[u1,u2]=meshgrid(linspace(-1.8,1.8,6),linspace(-1.8,1.8,6));
X_u = [u1(:) u2(:)];

% Initialize test points
[xt1,xt2]=meshgrid(-1.8:0.1:1.8,-1.8:0.1:1.8);
xt=[xt1(:) xt2(:)];

% set the data points into clusters. Here we construct two cell arrays.
%  trindex  contains the block index vectors for training data. That is
%           x(trindex{i},:) and y(trindex{i},:) belong to the i'th block.
%  tstindex contains the block index vectors for test data. That is test
%           inputs p(tstindex{i},:) belong to the i'th block.
%
b1 = [-1.7 -0.8 0.1 1 1.9];
mask = zeros(size(x,1),size(x,1));
trindex={}; tstindex={};
for i1=1:4
    for i2=1:4
        ind = 1:size(x,1);
        ind = ind(: , b1(i1)<=x(ind',1) & x(ind',1) < b1(i1+1));
        ind = ind(: , b1(i2)<=x(ind',2) & x(ind',2) < b1(i2+1));
        trindex{4*(i1-1)+i2} = ind';
        ind2 = 1:size(p,1);
        ind2 = ind2(: , b1(i1)<=p(ind2',1) & p(ind2',1) < b1(i1+1));
        ind2 = ind2(: , b1(i2)<=p(ind2',2) & p(ind2',2) < b1(i2+1));
        tstindex{4*(i1-1)+i2} = ind2';
    end
end

% Create the PIC GP data structure and set the inducing inputs and block indeces
gpcf = gpcf_sexp('lengthScale', [1 1], 'magnSigma2', 0.2^2);
lik = lik_gaussian('sigma2', 0.2^2);

gp_pic = gp_set('type', 'PIC', 'lik', lik, 'cf', {gpcf}, ...
                'X_u', X_u, 'tr_index', trindex, 'jitterSigma2', 1e-4);
```

Now the cell array `trindex` contains the block index vectors for the training data. It means that, for example, the inputs and outputs `x(trindex{i},:)` and `y(trindex{i},:)` belong to the i'th block. The optimization of parameters and inducing inputs is done the same way as with FIC or a full GP model. In prediction, however, we have to give one extra input, `tstindex`, for `gp_pred`. This defines how the prediction inputs are appointed in the blocks in a same manner as `trindex` appoints the training inputs.

```
Eft_pic = gp_pred(gp_pic, x, y, xt, 'tstind', tstindex);
```

Figure 4.2 shows the predicted surface. One should notice that the PIC's prediction is discontinuous whereas the prediction with FIC and full GP are continuous. The discontinuities take place in the block boundaries and are a result of discontinuous covariance function that PIC resembles. This issue is discussed in more detail by Vanhatalo et al. (2010).

4.3 Deterministic training conditional, subset of regressors and variational sparse approximation

The deterministic training conditional is based on the works by Csató and Opper (2002) and Seeger et al. (2003) and is earlier called Projected Latent Variables (see Quiñonero-

Candela and Rasmussen, 2005, for more details). The approximation can be constructed similarly as FIC and PIC by defining the inducing conditional, which in the case of DTC is

$$q(\mathbf{f}\,|\mathbf{X}, \mathbf{X}_u, \mathbf{u}, \theta) = \mathrm{N}(\mathbf{f}\,|\,\mathbf{K}_{\mathrm{f,u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{u}, 0). \tag{4.7}$$

This implies that the approximate prior over latent variables is

$$q(\mathbf{f}\,|\mathbf{X}, \mathbf{X}_u, \theta) = \mathrm{N}(\mathbf{f}\,|\mathbf{0}, \mathbf{K}_{\mathrm{f,u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{K}_{\mathrm{u,f}}). \tag{4.8}$$

The deterministic training conditional is not strictly speaking a proper GP since it uses different covariance function for the latent variables appointed to the training inputs and for the latent variables at the prediction sites, $\tilde{f}$. The prior covariance for $\tilde{f}$ is the true covariance $\mathbf{K}_{\tilde{\mathrm{f}},\tilde{\mathrm{f}}}$ instead of $\mathbf{K}_{\tilde{\mathrm{f}},\mathrm{u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{K}_{\mathrm{u},\tilde{\mathrm{f}}}$. This does not affect the predictive mean since the cross covariance $\mathrm{Cov}[\mathbf{f}, \tilde{\mathbf{f}}] = \mathbf{K}_{\mathrm{f,u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{K}_{\mathrm{u},\tilde{\mathrm{f}}}$, but it gives a larger predictive variance. An older version of DTC is the subset of regressors (SOR) sparse approximation which utilizes $\mathbf{K}_{\tilde{\mathrm{f}},\mathrm{u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{K}_{\mathrm{u},\tilde{\mathrm{f}}}$. However, this resembles a singular Gaussian distribution and thus the predictive variance may be negative. DTC tries to fix this problem by using $\mathbf{K}_{\tilde{\mathrm{f}},\tilde{\mathrm{f}}}$ (see Quiñonero-Candela and Rasmussen, 2005). DTC and SOR are identical in other respects than in the predictive variance evaluation. In spatial statistics, SOR has been used by Banerjee et al. (2008) with a name Gaussian predictive process model.

The approximate prior of the variational approximation by Titsias (2009) is exactly the same as that of DTC. The difference between the two approximations is that in variational setting the inducing inputs and covariance function parameters are optimized differently. The inducing inputs and parameters can be seen as variational parameters that should be chosen to maximize the variational lower bound between the true GP posterior and the sparse approximation. This leads to optimization of modified log marginal likelihood

$$V(\theta, \mathbf{X}_u) = \log[N(\mathbf{y}\,|0, \sigma^2\mathbf{I} + \mathbf{Q}_{\mathrm{f,f}})] - \frac{1}{2\sigma^2}\mathrm{tr}(\mathbf{K}_{\mathrm{f,f}} - \mathbf{K}_{\mathrm{f,u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{K}_{\mathrm{u,f}}) \tag{4.9}$$

with Gaussian observation model. With non-Gaussian observation model, the variational lower bound is similar but $\sigma^2\mathbf{I}$ is replaced by $\mathbf{W}^{-1}$ (Laplace approximation) or $\tilde{\mathbf{\Sigma}}$ (EP).

Variational, DTC and SOR sparse approximation in GPstuff

The variational, DTC and SOR sparse approximations are constructed similarly to FIC. Only the type of the GP changes:

```
gp_var = gp_set('type', 'VAR', 'lik', lik, 'cf', {gpcf}, 'X_u', X_u);
gp_dtc = gp_set('type', 'DTC', 'lik', lik, 'cf', {gpcf}, 'X_u', X_u);
gp_sor = gp_set('type', 'SOR', 'lik', lik, 'cf', {gpcf}, 'X_u', X_u);
```

4.3.1 Comparison of sparse GP models

Figure 4.2 shows the predictive mean of all the sparse approximations (the mean of SOR is the same as that of DTC). It should be noticed that variational approximation is closest to the full GP solution in Figure 3.2. The FIC and PIC approximations are most close to the full solution. FIC works rather differently on one corner of the region whereas the latent surface predicted by PIC contains discontinuities. DTC suffers most on the borders of the region.

The sparse GP approximations are compared also in the demo `demo_regression_sparse2`, which demonstrates the differences between FIC, variational and DTC in other context. See also the discussions on the differences between these sparse approximations given by (Quiñonero-Candela and Rasmussen, 2005; Snelson, 2007; Titsias, 2009; Alvarez et al., 2010).

4.3.2 Sparse GP models with non-Gaussian likelihoods

The extension of sparse GP models to non-Gaussian likelihoods is very straightforward in GPstuff. User can define the sparse GP just as described in the previous two sections and then continue with the construction of likelihood exactly the same way as with a full GP. The Laplace approximation, EP and integration methods can be used with the same commands as with full GP. This is demonstrated in `demo_spatial1`.

Chapter 5

Model assessment and comparison

There are various means to assess the goodness of the model and its predictive performance and GPstuff provides built in functionalities to many common test statistics. In this chapter, we will briefly discuss the model comparison and assesment in general and introduce a few basic methods that can be conducted routinely with GPstuff's tools.

5.1 Marginal likelihood

Marginal likelihood is often used for model selection (see, e.g. Kass and Raftery, 1995). It corresponds to ML II or with model priors to MAP II estimate in the model space, selecting the model with the highest marginal likelihood or highest marginal posterior probability. The use of this in model selection is as justified as using a MAP point estimate for the parameters. It works well if the posterior is concentrated to a single model, that is, if single model produce similar predictions as Bayesian model avarage model obtained by integrating over the model space.

In GPstuff, if MAP estimate and integration approximation (IA) are almost the same, marginal likelihood can be used as a quick estimate to compare models, but we recommend using cross-validation for more thorough model assessment and selection.

5.2 Predictive performance estimates

In prediction problems it is natural to assess the predictive performance of the model by focusing on the model's predictive distribution (Good, 1952; Bernardo and Smith, 2000). The posterior predictive distribution of an output y_{n+1} given the new input $\mathbf{x}_{n+1}$ and the training data $\mathcal{D} = \{(\mathbf{x}_i, y_i); i = 1, 2, \ldots, n\}$ is obtained by marginalizing over the unknown latent variable and parameters θ given the model M

$$p(y_{n+1}|\,\mathbf{x}_{n+1}, \mathcal{D}, M) = \int p(y_{n+1}|\,\mathbf{x}_{n+1}, \theta, \phi, \mathcal{D}, M) p(\theta|\,\mathbf{x}_{n+1}, \mathcal{D}, M) d\theta d\phi, \quad (5.1)$$

where $p(y_{n+1}|\,\mathbf{x}_{n+1}, \theta, \phi, \mathcal{D}, M) = \int p(y_{n+1}|f_{n+1}, \phi) p(f_{n+1}|\,\mathbf{x}_{n+1}, \theta, \mathcal{D}, M) df_{n+1}$. In the following, we will assume that knowing $\mathbf{x}_{n+1}$ does not give more information about θ or ϕ, that is, $p(\theta, \phi|\,\mathbf{x}_{n+1}, \mathcal{D}, M) = p(\theta, \phi|\mathcal{D}, M)$.

To estimate the predictive performance of the model we would like to compare the posterior predictive distribution to future observations from the same process that generated the given set of training data $\mathcal{D}$. Agreement or discrepancy between the predictive distribution and the observations can be measured with a utility or loss function, $u\left(y_{n+1}, \mathbf{x}_{n+1}, \mathcal{D}, M\right)$. Preferably, the utility u would be application-specific, measuring the expected benefit (or cost) of using the model. Good generic utility function is the log-score which is called log predictive likelihood, $\log p(y_{n+1}|\, \mathbf{x}_{n+1}, \mathcal{D}, M)$, when used for predictive density. It measures how well the model estimates the whole predictive distribution (Bernardo, 1979) and is thus especially useful in model comparison.

Usually since future observations are not yet available, we need to estimate the expected utility by taking the expectation over the future data distribution

$$\bar{u} = \mathrm{E}_{(\mathbf{x}_{n+1}, y_{n+1})}\left[u\left(y_{n+1}, \mathbf{x}_{n+1}, \mathcal{D}, M\right)\right]. \tag{5.2}$$

There are several methods for estimating (5.2). GPstuff provides two commonly used approaches: cross-validation and deviance information criterion.

5.3 Cross-validation

Cross-validation (CV) is an approach to compute predictive performance estimate by re-using observed data. As the distribution of $(\mathbf{x}_{n+1}, y_{n+1})$ is unknown, we assume that it can be reasonably well approximated using the training data $\{(\mathbf{x}_i, y_i); i = 1, 2, \ldots, n\}$. To avoid the double conditioning on the training data and simulate the fact that the future observations are not in the training data, the ith observation $(\mathbf{x}_i, y_i)$ in the training data is left out, and then the predictive distribution for y_i is computed with a model that is fitted to all of the observations except $(\mathbf{x}_i, y_i)$. By repeating this for every point in the training data, we get a collection of leave-one-out cross-validation (LOO-CV) predictive densities

$$\{p(y_i|\, \mathbf{x}_i, \mathcal{D}_{\setminus i}, M); i = 1, 2, \ldots, n\}, \tag{5.3}$$

where $\mathcal{D}_{\setminus i}$ denotes all the elements of $\mathcal{D}$ except $(\mathbf{x}_i, y_i)$. To get the expected utility estimate, these predictive densities are compared to the actual y_i's using the utility u, and the expectation is taken over i

$$\bar{u}_{\mathrm{LOO}} = \mathrm{E}_i\left[u(y_i, \mathbf{x}_i, \mathcal{D}_{\setminus i}, M)\right]. \tag{5.4}$$

The right hand side terms are conditioned on $n - 1$ data points, making the estimate almost unbiased.

5.3.1 Leave-one-out cross-validation

For GP with a Gaussian noise model and given covariance parameters, the LOO-CV predictions can be computed using an analytical solution (Sundararajan and Keerthi, 2001), which is implemented in `gp_loopred`.

5.3.2 k-fold cross-validation

The LOO-CV is computationally feasible only for the Gaussian likelihood and fixed parameters with `gp_looe`. To reduce computation time, in k-fold-CV, we use only k

(e.g. $k = 10$) k-fold-CV distributions $p(\theta|D_{(\setminus s(i))}, M)$ and get a collection of predictive densities

$$\{p(y_i|\,\mathbf{x}_i, \mathcal{D}_{\setminus s(i)}, M); i = 1, 2, \ldots, n\}, \tag{5.5}$$

where $s(i)$ is a set of data points as follows: the data are divided into k groups so that their sizes are as nearly equal as possible and $s(i)$ is the set of data points in the group where the ith data point belongs. The expected utility estimated by the k-fold-CV is then

$$\bar{u}_{\mathrm{CV}} = \mathrm{E}_i\left[u(y_i, \mathbf{x}_i, \mathcal{D}_{\setminus s(i)}, M)\right]. \tag{5.6}$$

Since the k-fold-CV predictive densities are based on smaller training data sets $\mathcal{D}_{\setminus s(i)}$ than the full data set $\mathcal{D}$, the expected utility estimate is slightly biased. This bias can be corrected using a first order bias correction (Burman, 1989):

$$\bar{u}_{\mathrm{tr}} = \mathrm{E}_i[u(y_i, \mathbf{x}_i, \mathcal{D}, M)] \tag{5.7}$$

$$\bar{u}_{\mathrm{cvtr}} = \mathrm{E}_j\left[\mathrm{E}_i[u(y_i, \mathbf{x}_i, \mathcal{D}_{\setminus s_j}, M)]\right] \quad ; \quad j = 1, \ldots, k \tag{5.8}$$

$$\bar{u}_{\mathrm{CCV}} = \bar{u}_{\mathrm{CV}} + \bar{u}_{\mathrm{tr}} - \bar{u}_{\mathrm{cvtr}}, \tag{5.9}$$

where $\bar{u}_{\mathrm{tr}}$ is the expected utility evaluated with the training data given the training data and $\bar{u}_{\mathrm{cvtr}}$ is the average of the expected utilities evaluated with the training data given the k-fold-CV training sets.

GPstuff provides `gp_kfcv`, which computes k-fold-CV and bias-corrected k-fold-CV with log-score and root mean squared error (RMSE). The function `gp_kfcv` provides also basic variance estimates for the predictive performance estimates. First the mean expected utility $\bar{u}_j$ for each k folds is computed. $\bar{u}_j$'s tend to be closer to Gaussian (due to the central limit theorem) and then the variance of the expected utility is computed as (see, e.g., Dietterich, 1998)

$$\mathrm{Var}[\bar{u}] \approx \mathrm{Var}_j[\bar{u}_j]/k. \tag{5.10}$$

Although some information is lost by first taking the sub-expectations, the estimate is useful indicator of the related uncertainty. See (Vehtari and Lampinen, 2002) for more details on estimating the uncertainty in performance estimates.

5.4 DIC

Deviance information criterion (DIC) is another very popular model selection criterion (Spiegelhalter et al., 2002). DIC estimates aslo the predictive performance, but replaces the predictive distribution with plug-in predictive distribution, where plug-in estimate $\hat{\theta}$ is used, and uses deviance as the loss function. With parametric models without any hierarchy it is usually written as

$$p_{\mathrm{eff}} = \mathrm{E}_{\theta|\mathcal{D}}[D(\mathbf{y}, \theta)] - D(\mathbf{y}, \mathrm{E}_{\theta|\mathcal{D}}[\theta]) \tag{5.11}$$

$$\mathrm{DIC} = \mathrm{E}_{\theta|\mathcal{D}}[D(\mathbf{y}, \theta)] + p_{\mathrm{eff}}, \tag{5.12}$$

where p_{eff} is the effective number of parameters and $D = -2\log(p(\mathbf{y}\,|\theta))$ is the deviance. Since our models are hierarchical we need to decide the parameters on focus (see Spiegelhalter et al., 2002, for discussion on this). The parameters on the focus

are those over which the expectations are taken when evaluating the effective number of parameters and DIC. In the above equations, the focus is in the parameters and in the case of a hierarchical GP model of GPstuff the latent variables would be integrated out before evaluating DIC. If we have a MAP estimate for the parameters, we may be interested to evaluate DIC statistics with the focus on the latent variables. In this case the above formulation would be

$$p_D(\theta) = \mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta}[D(\mathbf{y},\mathbf{f})] - D(\mathbf{y},\mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta}[\mathbf{f}]) \tag{5.13}$$

$$\mathrm{DIC} = \mathrm{E}_{\mathbf{f}\,|\mathcal{D},\theta}[D(\mathbf{y},\mathbf{f})] + p_D(\theta). \tag{5.14}$$

Here the effective number of parameters is denoted differently with $p_D(\theta)$ since now we are approximating the effective number of parameters in $\mathbf{f}$ conditionally on θ, which is different from the p_{eff}. $p_D(\theta)$ is a function of the parameters and it measures to what extent the prior correlations are preserved in the posterior of the latent variables given θ. For non-informative data $p_D(\theta) = 0$ and the posterior is the same as the prior. The greater $p_D(\theta)$ is the more the model is fitted to the data and large values compared to n indicate potential overfit. Also, large $p_D(\theta)$ indicates that we cannot assume that the conditional posterior approaches normality by central limit theorem. Thus, $p_D(\theta)$ can be used for assessing the goodness of the Laplace or EP approximation for the conditional posterior of the latent variables as discussed by Rue et al. (2009) and Vanhatalo et al. (2010). The third option is to evaluate DIC with focus on all the variables, $[\mathbf{f}, \theta]$. In this case the expectations are over $p(\mathbf{f}, \theta|\mathcal{D})$.

5.5 Model assessment demos

The model assessment methods are demonstrated with the functions `demo_modelassessmentt1` and `demo_modelassessment2`. The former compares the sparse GP approximations to the full GP with regression data and the latter compares the logit and probit likelihoods in GP classification.

5.5.1 demo_modelassessment1

Assume that we have built our regression model with a Gaussian noise and used optimization method to find the MAP estimate for the parameters. We evaluate the effective number of latent variables and DIC statistics.

```
p_eff_latent = gp_peff(gp, x, y);
[DIC_latent, p_eff_latent2] = gp_dic(gp, x, y, 'focus', 'latent');
```

where p_{eff} is evaluated with two different approximations. Since we have the MAP estimate for the parameters the focus is on the latent variables. In this case we can also use `gp_peff` which returns the effective number of parameters approximated as

$$p_D(\theta) \approx n - \mathrm{tr}(\mathbf{K}_{\mathrm{f,f}}^{-1}(\mathbf{K}_{\mathrm{f,f}}^{-1} + \sigma_n^{-2}\mathbf{I})^{-1}) \tag{5.15}$$

(Spiegelhalter et al., 2002). When the focus is on the latent variables, the function `gp_dic` evaluates the DIC statistics and the effective number of parameters as described by the equations (5.13) and (5.14). The k-fold-CV expected utility estimate can be evaluated as follows.

```
cvres =  gp_kfcv(gp, x, y);
```

The gp_kfcv takes the ready made model structure gp and the training data x and y. The function divides the data into k groups, conducts inference separately for each of the training groups and evaluates the expected utilities with the test groups. Since no optional parameters are given the inference is conducted using MAP estimate for the parameters. The default division of the data is into 10 groups. The expected utilities and their variance estimates are stored in the structure cvres as follows:

```
cvres =
        mlpd_cv: 0.0500
     Var_lpd_cv: 0.0014
       mrmse_cv: 0.2361
    Var_rmse_cv: 1.4766e-04
        mabs_cv: 0.1922
     Var_abs_cv: 8.3551e-05
```

gp_kfcv returns also other statistics if more information is needed and the function can be used to save the results automatically. However, these functionalities are not considered here. Readers interested on detailed analysis should read the help text for gp_kfcv.

Now we will turn our attention to other inference methods than MAP estimate for the parameters. Assume we have a record structure from gp_mc function with Markov chain samples of the parameters stored in it. In this case, we have two options how to evaluate the DIC statistics. We can set the focus on the parameters or all the parameters (that is parameters and latent variables). The two versions of DIC and effective number of parameters are evaluated as follows:

```
rgp = gp_mc(gp, x, y, opt);
[DIC, p_eff] =  gp_dic(rgp, x, y, 'focus', 'param');
[DIC2, p_eff2] =  gp_dic(rgp, x, y, 'focus', 'all');
```

Here the first line performs the MCMC sampling with options opt. The last two lines evaluate the DIC statistics. With Markov chain samples, we cannot use the gp_peff function to evaluate $p_D(\theta)$ since that is a special function for models with fixed parameters. The k-fold-CV is conducted with MCMC methods as easily as with the MAP estimate. The only difference is that we have to define that we want to use MCMC and to give the sampling options for gp_kfcv. These steps are done as follows:

```
opt.nsamples= 100;
opt.repeat=4;
opt.hmc_opt = hmc2_opt;
opt.hmc_opt.steps=4;
opt.hmc_opt.stepadj=0.05;
opt.hmc_opt.persistence=0;
opt.hmc_opt.decay=0.6;
opt.hmc_opt.nsamples=1;
hmc2('state', sum(100*clock));

cvres =  gp_kfcv(gp, x, y, 'inf_method', 'MCMC', 'opt', opt);
```

With integration approximation evaluating the DIC and k-fold-CV statistics are similar to the MCMC approach. The steps required are:

```
opt.int_method = 'grid';
opt.step_size = 2;
opt.optimf=@fminscg;

gp_array = gp_ia(gp, x, y, opt);

models{3} = 'full_IA';
[DIC(3), p_eff(3)] =  gp_dic(gp_array, x, y, 'focus', 'param');
[DIC2(3), p_eff2(3)] =  gp_dic(gp_array, x, y, 'focus', 'all');

% Then the 10 fold cross-validation.
cvres = gp_kfcv(gp, x, y, 'inf_method', 'IA', 'opt', opt);
```

This far we have demonstrated how to use DIC and k-fold-CV functions with full GP. The function can be used with sparse approximations exactly the same way as with full GP and this is demonstrated in demo_modelassessment1 for FIC and PIC.

5.5.2 demo_modelassessment2

The functions `gp_peff`, `gp_dic` and `gp_kfcv` work similarly for non-Gaussian likelihoods as for a Gaussian one. The only difference is that the integration over the latent variables is done approximately. The way the latent variables are treated is defined in the field `latent_method` of the GP structure and this is initialized when constructing the model as discussed in the section 3.2. If we have conducted the analysis with Laplace approximation and MAP estimate for the parameters, we can evaluate the DIC and k-fold-CV statistics as follows:

```
p_eff_latent = gp_peff(gp, x, y);
[DIC_latent, p_eff_latent2] = gp_dic(gp, x, y, 'focus', 'latent');

% Evaluate the 10-fold cross validation results.
cvres = gp_kfcv(gp, x, y);
```

These are exactly the same lines as presented earlier with the Gaussian likelihood. The difference is that the GP structure `gp` and the data `x` and `y` are different and the integrations over latent variables in `gp_dic` and `gp_kfcv` are done with respect to the approximate conditional posterior $q(\mathbf{f}\,|\hat{\theta}, \mathcal{D})$ (which was assumed to be Laplace approximation). The effective number of parameters returned by `gp_peff` is evaluated as in the equation (5.15) with the modification that $\sigma_n^{-2}\mathbf{I}$ is replaced by $\mathbf{W}$ in the case of Laplace approximation and $\tilde{\mathbf{\Sigma}}^{-1}$ in the case of EP.

If expectation propagation is used for inference the model assessment is conducted similarly as with Laplace approximation. Also the MCMC and IA solutions are evaluated identically to the Gaussian case. For this reason the code is not repeated here.

Chapter 6

Covariance functions

In the previous chapters we have not paid much attention on the choice of the covariance function. However, GPstuff has rather versatile collection of covariance functions, which can be combined in numerous ways. The different functions are collected in the Appendix B. This chapter demonstrates some of the functions and ways to combine them.

6.1 Neural network covariance function

A good example of covariance function that has very different properties than the standard stationary covariance functions such as squared exponential or M?ern covariance functions is the neural network covariance function. In this section we will demonstrate its use in two simple regression problems. The squared exponential covariance function is taken as a reference and the code is found from the `demo_neuralnetCov`.

6.2 Additive models

In many practical situations, a GP prior with only one covariance function may be too restrictive since such a construction can model effectively only one phenomenon. For example, the latent function may vary rather smoothly across the whole area of interest, but at the same time it can have fast local variations. In this case, a more reasonable model would be

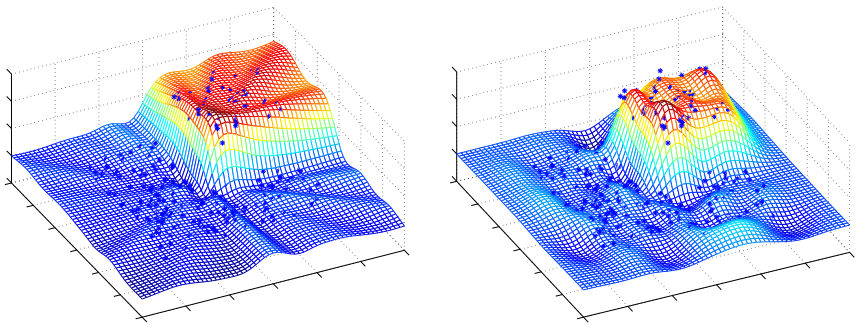
$$f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x}), \tag{6.1}$$

where the latent value function is a sum of two functions, slow and fast varying. By placing a separate GP prior for both of the functions g and h we obtain an additive prior

$$f(\mathbf{x})|\theta \sim \mathcal{GP}(0, k_g(\mathbf{x}, \mathbf{x}') + k_h(\mathbf{x}, \mathbf{x}')). \tag{6.2}$$

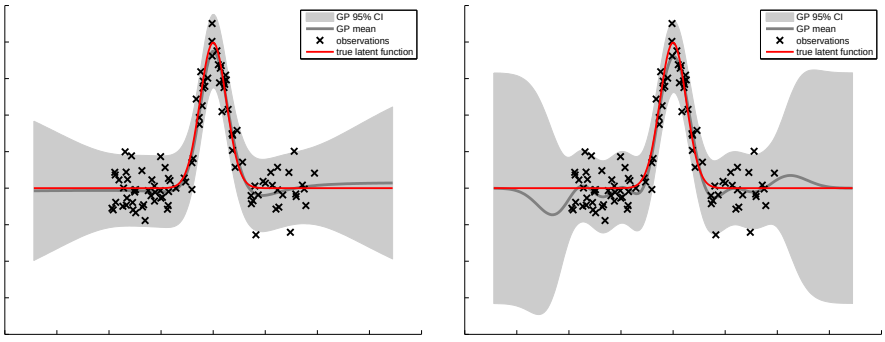
The marginal likelihood and posterior distribution of the latent variables are as before with $\mathbf{K}_{\mathrm{f,f}} = \mathbf{K}_{\mathrm{g,g}} + \mathbf{K}_{\mathrm{h,h}}$. However, if we are interested on only, say, phenomenon g, we can consider the h part of the latent function as correlated noise and evaluate the predictive distribution for g, which with the Gaussian observation model would be

$$\tilde{g}(\tilde{\mathbf{x}})|\mathcal{D}, \theta \sim \mathcal{GP}\left(k_g(\tilde{\mathbf{x}}, \mathbf{X})(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}\mathbf{y}, k_g(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') - k_g(\tilde{\mathbf{x}}, \mathbf{X})(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}k_g(\mathbf{X}, \tilde{\mathbf{x}}')\right). \tag{6.3}$$



(a) Neural network solution. (b) Squared exponential solution.

Figure 6.1: GP latent mean predictions (using a MAP estimate) with neural network or squared exponential covariance functions. The 2D data is generated from a step function.



(a) Neural network solution. (b) Squared exponential solution.

Figure 6.2: GP solutions (a MAP estimate) with neural network or squared exponential covariance functions.

With non-Gaussian likelihood, the Laplace and EP approximations for this are similar since only $\sigma^2\mathbf{I}$ and $(\mathbf{K}_{\mathrm{f,f}} + \sigma^2\mathbf{I})^{-1}\mathbf{y}$ change in the approximations.

The multiple length-scale model can be formed also using specific covariance functions. For example, A rational quadratic covariance function (`gpcf_rq`) can be seen as a scale mixture of squared exponential covariance functions (Rasmussen and Williams, 2006), and could be useful for data that contain both local and global phenomena. However, using sparse approximations with the rational quadratic would prevent it from modeling local phenomena . The additive model (6.2) suits better for sparse GP formalism since it enables to combine FIC with CS covariance functions.

As discussed in section 4.2, FIC can be interpreted as a realization of a special kind of covariance function. By adding FIC with CS covariance function, for example (4.1), one can construct a sparse additive GP prior which implies the latent variable prior

$$\mathbf{f} \mid \mathbf{X}, \mathbf{X}_u, \theta \sim \mathrm{N}(\mathbf{0}, \mathbf{K}_{\mathrm{f,u}}\,\mathbf{K}_{\mathrm{u,u}}^{-1}\,\mathbf{K}_{\mathrm{u,f}} + \hat{\mathbf{\Lambda}}). \tag{6.4}$$

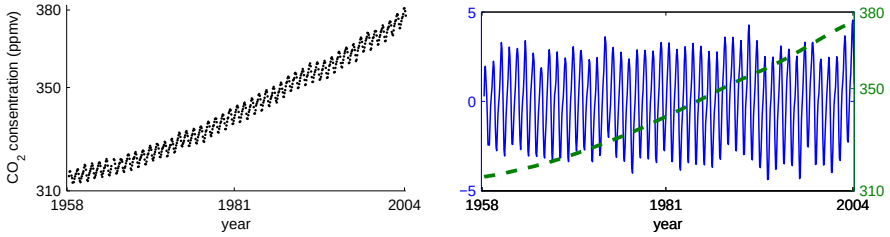
This prior will be referred as CS+FIC. Here, the matrix $\hat{\mathbf{\Lambda}} = \mathbf{\Lambda} + k_{\mathrm{pp,q}}(\mathbf{X}, \mathbf{X})$ is sparse with the same sparsity structure as in $k_{\mathrm{pp,q}}(\mathbf{X}, \mathbf{X})$ and it is fast to use in computations and cheap to store. CS+FIC can be extended to have more than one component. However, it should be remembered that FIC works well only for long length-scale phenomena and the computational benefits of CS functions are lost if their length-scale gets too large (Vanhatalo et al., 2010). For this reason the CS+FIC should be constructed so that possible long length-scale phenomena are handled with FIC part and the short length-scale phenomena with CS part. The implementation of the CS+FIC model follows closely the implementation of FIC and PIC (for details see Vanhatalo and Vehtari, 2008, 2010).

In the following sections we will demonstrate the additive models with two problems. First we will consider full GP with covariance function that is a sum of periodic and squared exponential covariance function. This GP prior is demonstrated for a Gaussian and non-Gaussian likelihood. The second demo concentrates on sparse GPs in additive models. The FIC, PIC and CS+FIC sparse models are demonstrated with data set that contains both long and short length-scale phenomena.

6.2.1 Additive models demo: demo_periodic

In this section we will discuss the demonstration program `demo_periodic`. This demonstrates the use of a periodic covariance function `gpcf_periodic` with two data sets, the Mauna Loa CO2 data (see, for example, Rasmussen and Williams, 2006) and the monthly Finnish drowning statistics 2002-2008. The first data is a regression problem with Gaussian noise whereas the second consist of count data that is modeled with Poisson observation model. Here, we will describe only the regression problem the other data can be examined by running the demo.

We will analyze the Mauna Loa CO2 data with two additive models. The first one utilizes covariance function that is a sum of squared exponential and piece-wise polynomial $k_{\mathrm{se}}(\mathbf{x}, \mathbf{x}') + k_{\mathrm{pp},2}(\mathbf{x}, \mathbf{x}')$. The solution of this model that shows the long and short length-scale phenomena separately is visualized in Figure 6.3 together with the original data. This model interpolates the underlying function well but as will be demonstrated later its predictive properties into the future are not so good. Better predictive performance is obtained by adding up two squared exponential and one periodic covariance function $k_{\mathrm{se}}(\mathbf{x}, \mathbf{x}' \,|\theta_1) + k_{\mathrm{se}}(\mathbf{x}, \mathbf{x}' \,|\theta_2) + k_{\mathrm{periodic}}(\mathbf{x}, \mathbf{x}')$, which is build as follows.



(a) The Mauna Loa CO_2 data.

(b) The long and short (left scale) term component (right scale).

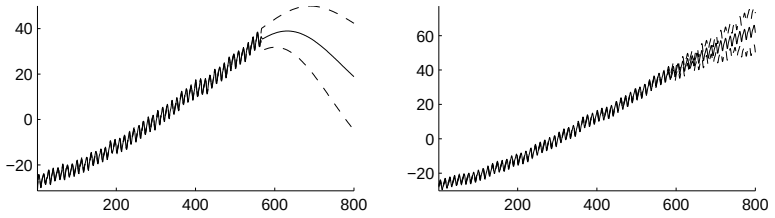
Figure 6.3: The Mauna Loa CO_2 data and prediction for the long term and short term component demo_regression2.

```
gpcf1 = gpcf_sexp('lengthScale', 67*12, 'magnSigma2', 66*66);
gpcfp = gpcf_periodic('lengthScale', 1.3, 'magnSigma2', 2.4*2.4);
gpcfp = gpcf_periodic(gpcfp, 'period', 12,'optimPeriod',1,'lengthScale_exp', 90*12, 'decay', 1);
lik = lik_gaussian('sigma2', 0.3);
gpcf2 = gpcf_sexp('lengthScale', 2, 'magnSigma2', 2);
pl = prior_t('s2', 10, 'nu', 3);
pn = prior_t('s2', 10, 'nu', 4);

gpcf1 = gpcf_sexp(gpcf1, 'lengthScale_prior', pl, 'magnSigma2_prior', pl);
gpcf2 = gpcf_sexp(gpcf2, 'lengthScale_prior', pl, 'magnSigma2_prior', pl);
gpcfp = gpcf_periodic(gpcfp, 'lengthScale_prior', pl, 'magnSigma2_prior', pl);
gpcfp = gpcf_periodic(gpcfp, 'lengthScale_sexp_prior', pl, 'period_prior', pn);
lik = lik_gaussian(lik, 'sigma2_prior', pn);

gp = gp_set('lik', lik, 'cf', {gpcf1, gpcfp, gpcf2})
```

An additive model is constructed similarly to a model with just one covariance function. The only difference is that now we give more than one covariance function structure for the `gp_init`. The inference with additive model is conducted exactly the same way as in the `demo_regression1`. The below lines summarize the hyperparameter optimization and conduct the prediction for the whole process f?and two components g and h whose covariance functions are $k_{\text{se}}(\mathbf{x}, \mathbf{x}' \,|\theta_1)$ and $k_{\text{se}}(\mathbf{x}, \mathbf{x}' \,|\theta_2) + k_{\text{periodic}}(\mathbf{x}, \mathbf{x}')$ respectively. The prediction for latent functions that are related to only subset of the covariance functions used for training is done by setting an optional argument `'predcf'`. This argument tells which covariance functions are used for the prediction. If we want to use more than one covariance function for prediction, and there are more than two covariance functions in the model, the value for `'predcf'` is a vector.

```
opt = optimset('TolFun', 1e-3, 'TolX', 1e-3, 'Display', 'iter');
gp = gp_optim(gp,x,y,'opt',opt);
xt=[1:800]';
[Ef_full, Varf_full, Ey_full, Vary_full] = gp_pred(gp, x, y, xt);
[Ef_full1, Varf_full1] = gp_pred(gp, x, y, x, 'predcf', 1);
[Ef_full2, Varf_full2] = gp_pred(gp, x, y, x, 'predcf', [2 3]);
```

The two components `Ef_full1` and `Ef_full2` above are basically identical to the component shown in Figure 6.3, which shows that there is no practical difference in the interpolation performance between the two models considered in this demo. However, the additive model with periodic component has much more predictive power



(a) Two additive components.

(b) Three additive components with periodic covariance function.

Figure 6.4: The Mauna Loa CO_2 data. Prediction with two different models. On the left model with covariance function $k_{\text{se}}(\mathbf{x}, \mathbf{x}') + k_{\text{pp},2}(\mathbf{x}, \mathbf{x}')$ and on the right a model with covariance function $k_{\text{se}}(\mathbf{x}, \mathbf{x}' \,|\theta_1) + k_{\text{se}}(\mathbf{x}, \mathbf{x}' \,|\theta_2) + k_{\text{periodic}}(\mathbf{x}, \mathbf{x}')$. It can be seen that the latter has more predictive power.

into the future. This is illustrated in Figure 6.4 where one can see that the prediction with non-periodic model starts to decrease towards prior mean very quickly, and does not extrapolate the period, whereas the periodic model extrapolates the almost linear increase and periodic behaviour. The MCMC or grid integration approach for the additive model is identical to regression with only one covariance function and is not repeated here.

6.2.2 Additive models with sparse approximations

The Mauna Loa CO2 data set is studied with sparse additive Gaussian processes in the demo `demo_regression2`. There the covariance is $k_{\text{se}}(\mathbf{x}, \mathbf{x}') + k_{\text{pp},2}(\mathbf{x}, \mathbf{x}')$ since periodic covariance does not work well with sparse approximations. The model construction and inference is conducted similarly as in the previous section so we will not repeat it here. However, it is worth mentioning few things that should be noticed when running the demo.

PIC works rather well for this data set whereas FIC fails to recover the fast varying phenomenon. The reason for this is that the inducing inputs are too sparsely located so that FIC can not reveal the short length-scale phenomenon. In general, FIC is able to model only phenomena whose length-scale is long enough compared to the average distance between adjacent inducing inputs (see Vanhatalo et al., 2010, for details). PIC on the other hand is able to model also fast varying phenomena inside the blocks. Its drawback, however, is that the correlation structure is discontinuous which may result in discontinuous predictions. The CS+FIC model corrects these deficiencies.

In FIC and PIC the inducing inputs are parameters of every covariance function, which means that all the correlations are circulated through the inducing inputs and the shortest length-scale the GP is able to model is defined by the locations of the inducing inputs. The CS+FIC sparse GP is build differently. There the CS covariance function does not utilize inducing inputs but evaluates the correlations exactly. This enables the GP model to capture both the long and short length-scale phenomena. The GPstuff package is coded so that if the GP structure is defined to be CS+FIC all the CS functions are treated outside FIC approximation. Thus, the CS+FIC model requires that there is at least one CS covariance function and one globally supported function (such as squared exponential). If there are more than two covariance functions in the

GP structure all the globally supported functions utilize inducing inputs and all the CS functions are added to $\hat{\Lambda}$

6.3 Additive covariance functions with selected variables

In the demo (`demo_regression_additive`), we demonstrate how covariance functions can be modified so that they are functions of only a subset of inputs. We will consider modelling an artificial 2D regression data with additive covariance functions where the individual covariance functions use only either the first or second input variable. That is the covariance is $k_1(x_1, x_1'|\theta_1) + k_2(x_2, x_2'|\theta_2)$, where the covariance functions are of type $k_1(x_1, x_1'|\theta_1) : \Re \mapsto \Re$ instead of $k(\mathbf{x}, \mathbf{x}'|\theta) : \Re^2 \mapsto \Re$, which has been the usual case in previous demos. Remember the notation $\mathbf{x} = [x_1, ..., x_d]^{\mathrm{T}}$ and $\mathbf{x}' = [x_1', ..., x_d']^{\mathrm{T}}$. Also solutions from the covariance function that uses both input variables are shown for comparison. In the regression we assume a Gaussian noise. The hyperparameter values are set to their MAP estimate. The models considered in this demo utilize the following six covariance functions:

- constant and linear
- constant and squared exponential for the first input and linear for the second input
- squared exponential for the first input and squared exponential for the second input
- squared exponential
- neural network for the first input and neural network for the second input
- neural network.

We will demonstrate how to construct the first, second and fifth model.

A linear covariance function with constant term can be constructed in GPstuff as

```
% constant covariance function
gpcf_c = gpcf_constant('constSigma2', 1);
gpcf_c = gpcf_constant(gpcf_c, 'constSigma2_prior', pt);

% linear covariance function
gpcf_l = gpcf_linear('coeffSigma2_prior', pt);
```

Gaussian process using this linear covariance function is constructed as previously

```
gp = gp_set('lik', lik, 'cf', {gpcf_c gpcf_l}, 'jitterSigma2', jitter);
```

In this model, the covariance function is $c + k_{\text{linear}}(\mathbf{x}, \mathbf{x}'|\theta), \mathbf{x} \in \Re^2$, which means that the components of $\mathbf{x}$ are coupled in the in the covariance function k_{linear}.. The constant term (`gpcf_const`) is denoted by c.

The second model is more flexible. It contains a squared exponential, which is a function of the first input dimension x_1, and a linear covariance function, which is a function of the second input dimension x_2. The additive covariance function is $k_{\text{se}}(x_1, x_1'|\theta_1) + k_{\text{linear}}(x_2, x_2'|\theta_2)$ which is a mapping from $\Re^2$ to $\Re$. With the squared exponential covariance function, the inputs to be used can be selected using a metric structure as follows:

```
% Covariance function for the first input variable
gpcf_s1 = gpcf_sexp('magnSigma2', 0.15, 'magnSigma2_prior', pt);
% create metric structure:
metric1 = metric_euclidean({[1]},'lengthScales',[0.5], 'lengthScales_prior', pt);
% set the metric to the covariance function structure:
gpcf_s1 = gpcf_sexp(gpcf_s1, 'metric', metric1);
```

Here we construct the covariance function structure just as before. We also set a prior structure for the magnitude `pt`. To modify squared exponential covariance function so that it depends only on a subset of inputs is done using a metric structure. In this example, we use `metric_euclidean` which allows user to group the inputs so that all the inputs in one group are appointed to the same length-scale. The metric structure has function handles which then evaluate, for example, the distance with this modified euclidean metric. For example, for inputs $\mathbf{x}, \mathbf{x}' \in \Re^4$ the modified distance could be

$$r = \sqrt{(x_1 - x_1')^2/l_1^2 + \left((x_2 - x_2')^2 + (x_3 - x_3')^2\right)/l_2^2 + (x_4 - x_4')^2/l_3^2} \qquad (6.5)$$

where the second and third input dimension are given the same length-scale. This is different from the previously used, $r = \sqrt{\sum_{i=1}^{d}(x_i - x_i')^2/l_i^2}$ and $r = \sqrt{\sum_{i=1}^{d}(x_i - x_i')^2/l^2}$, that can be defined by the covariance function itself. The metric structures can be used with any stationary covariance function, that is with functions of type $k(\mathbf{x}, \mathbf{x}') = k(r)$. The reason why this property is implemented by using one extra structure is that this way user does not need to modify the covariance function when redefining the distance. Only a new metric file needs to be created. It should be noticed, though, that not all metrics lead to positive definite covariances with all covariance functions. For example, the squared exponential $\sigma^2 \exp(-r^2)$ is not positive definite with metric induced by L_1 norm $r = \sum_{i=1}^{d} |\mathbf{x}_i - \mathbf{x}_i'|/l_i$ whereas the exponential $\sigma^2 \exp(-r)$ is.

A metric structure cannot be used with the linear or neural network covariance functions, since they are not stationary but a smaller set of inputs can be chosen by using the field *selectedVariables*. `selectedVariables` can be used also for stationary covariance functions as a shorthand to set a metric structure to use the selected variables. With metric is also possible to do more elaborate input models as discussed in next section.

In this demo, we select only the second input variable as

```
gpcf_l2 = gpcf_linear('selectedVariables', [2]);
gpcf_l2 = gpcf_linear(gpcf_l2, 'coeffSigma2_prior', pt);
```

The result with this model is shown in Figure 6.5(b).

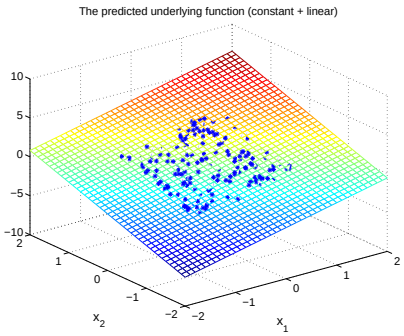
The neural network covariance function is another non-stationary covariance function with which the metric structure can not be used. However, a smaller set of input variables can be chosen similarly as with the linear covariance function using the field *selectedVariables*. In this demo, we consider additive neural network covariance functions, each having one input variable $k_{\text{nn}}(x_1, x_1'|\theta_1) + k_{\text{nn}}(x_2, x_2'|\theta_2)$. In GPstuff this can be done as

```
gpcf_nn1 = gpcf_neuralnetwork('weightSigma2', 1, 'biasSigma2', 1, 'selectedVariables', [1]);
gpcf_nn1 = gpcf_neuralnetwork(gpcf_nn1, 'weightSigma2_prior', pt, 'biasSigma2_prior', pt);

gpcf_nn2 = gpcf_neuralnetwork('weightSigma2', 1, 'biasSigma2', 1, 'selectedVariables', [2]);
gpcf_nn2 = gpcf_neuralnetwork(gpcf_nn2, 'weightSigma2_prior', pt, 'biasSigma2_prior', pt);

gp = gp_init('lik', lik, 'cf', {gpcf_nn1,gpcf_nn2}, jitterSigma2, jitter);
```

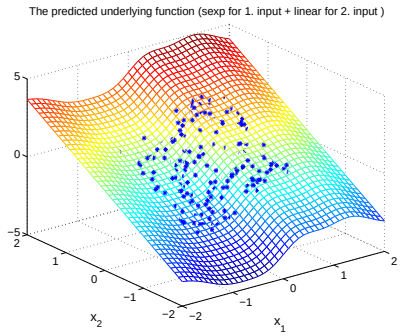
The result from this and other six models are shown in Figure 6.5.

6.4 Product of covariance functions

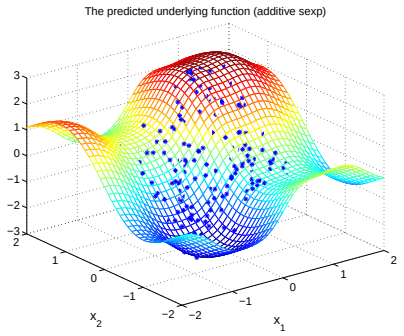
A product of two or more covariance functions is a valid covariance function as well. Such constructions may be useful in situations where the phenomenon is known to be separable. Combining covariance functions into product form $k_1(\mathbf{x}, \mathbf{x}') \cdot k_2(\mathbf{x}, \mathbf{x}')$... is straightforward in GPstuff. There is a special covariance function `gpcf_prod` for this purpose. For example, multiplying exponential and M?ern covariance functions is done as follows:



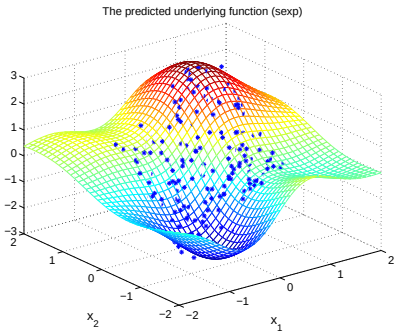
(a) $c + k_{\text{linear}}(\mathbf{x}, \mathbf{x}' | \theta)$



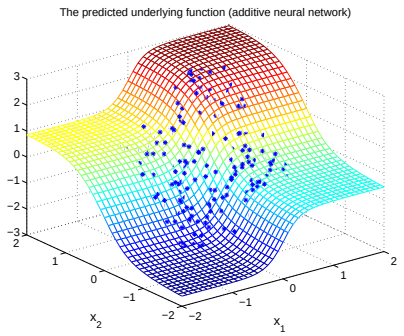
(b) $k_{\text{se}}(x_1, x_1' | \theta_1) + k_{\text{linear}}(x_2, x_2' | \theta_2)$



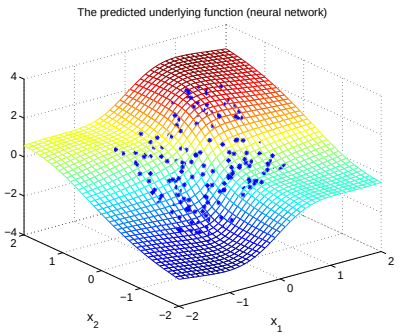
(c) $k_{\text{se}}(x_1, x_1' | \theta_1) + k_{\text{se}}(x_2, x_2' | \theta_2)$



(d) $k_{\text{se}}(\mathbf{x}, \mathbf{x}' | \theta_1)$



(e) $k_{\text{nn}}(x_1, x_1' | \theta_1) + k_{\text{nn}}(x_2, x_2' | \theta_2)$



(f) $k_{\text{nn}}(\mathbf{x}, \mathbf{x}' | \theta_1)$

Figure 6.5: GP latent mean predictions (using a MAP estimate) with different additive and non-additive covariance functions. The 2D toy data is generated from an additive process.

```
gpcf1 = gpcf_exp();
gpcf2 = gpcf_matern32();

gpcf = gpcf_prod('cf', {gpcf1, gpcf2});
```

Above we first initialized the two functions to be multiplied and in the third line we constructed a covariance function structure which handles the actual multiplication.

The product covariance can be combined with the metric structures also. For example, if we want to model a temporal component with one covariance function and the spatial components with other we can construct the covariance function $k(\mathbf{x}, \mathbf{x}') = k_1(x_1, x_1') \cdot k_2([x_2, x_3]^{\mathrm{T}}, [x_2', x_3']^{\mathrm{T}})$ as follows:

```
metric1 = metric_euclidean({[1]});
metric2 = metric_euclidean({[2 3]});
gpcf1 = gpcf_exp('metric', metric1);
gpcf2 = gpcf_matern32('metric', metric2);
gpcf = gpcf_prod('cf', {gpcf1, gpcf2});
```

The product covariance `gpcf_prod` can be used to combine categorical covariance `gpcf_cat` with other covariance functions to build hierarchical linear and non-linear models, as illustrated in `demo_regression_hier`. The above construction represents the covariance function

$$k(\mathbf{x}, \mathbf{x}') = k_{\mathrm{exp}}(x_1, x_1') \cdot k_{\mathrm{matern32}}([x_2, x_3]^{\mathrm{T}}, [x_2', x_3']^{\mathrm{T}}) \tag{6.6}$$

Chapter 7

Special observation models

In this chapter we will introduce few more models that we are able to infer with GP-stuff. These models utilize different observation models than what has been considered this far.

7.1 Robust regression with Student-t likelihood

A commonly used observation model in the GP regression is the Gaussian distribution. This is convenient since the inference is analytically tractable up to the covariance function parameters. However, a known limitation with the Gaussian observation model is its non-robustness, due which outlying observations may significantly reduce the accuracy of the inference. A formal definition of robustness is given, for example, in terms of an outlier-prone observation model. The observation model is outlier-prone of an order n, if $p(f|y_1, ..., y_{n+1}) \to p(f|y_1, ..., y_n)$ as $y_{n+1} \to \infty$ (O'Hagan, 1979; West, 1984). That is, the effect of a single conflicting observation on the posterior becomes asymptotically negligible as the observation approaches infinity. This contrasts heavily with the Gaussian observation model where each observation influences the posterior no matter how far it is from the others. A well-known robust observation model is the Student-t distribution

$$\mathbf{y} \mid \mathbf{f}, \nu, \sigma_t \sim \prod_{i=1}^{n} \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)\sqrt{\nu\pi}\sigma_t} \left(1 + \frac{(y_i - f_i)^2}{\nu\sigma_t^2}\right)^{-(\nu+1)/2}, \tag{7.1}$$

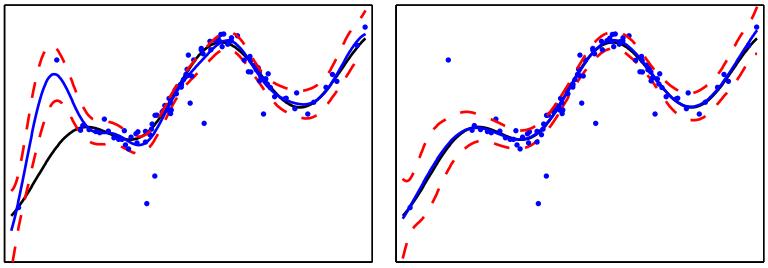
where ν is the degrees of freedom and σ the scale parameter (Gelman et al., 2004). Student-t distribution is outlier prone of order 1, and it can reject up to m outliers if there are at least $2m$ observations in all (O'Hagan, 1979).

The Student-t distribution can be utilized as such or it can be written via the scale mixture representation

$$y_i | f_i, \alpha, U_i \sim N(f_i, \alpha U_i) \tag{7.2}$$

$$U_i \sim \text{Inv-}\chi^2(\nu, \tau^2), \tag{7.3}$$

where each observation has its own noise variance αU_i that is Inv-χ^2 distributed (Neal, 1997; Gelman et al., 2004). The degrees of freedom ν corresponds to the degrees of freedom in the Student-t distribution and $\alpha\tau$ corresponds to σ.



(a) Gaussian observation model. (b) Student-t observation model.

Figure 7.1: An example of regression with outliers. On the left Gaussian and on the right the Student-t observation model. The real function is plotted with black line.

In GPstuff both of the representations are implemented. The scale mixture representation can be inferred only with MCMC and the Student-t observation model with Laplace approximation and MCMC.

7.1.1 Regression with Student-t distribution

Here we will discuss `demo_regression_robust`. The demo contains five parts: 1) Optimization approach with Gaussian noise, 2) MCMC approach with scale mixture noise model and all parameters sampled 3) Laplace approximation for Student-t likelihood optimizing all parameters, 4) MCMC approach with Student-t likelihood so that $\nu = 4$, and 5) Laplace approximation for Student-t observation model so that $\nu = 4$. We will demonstrate the steps for parts 2, 3 and 5.

The scale mixture model

The scale mixture representation of Student-t observation model is implemented as a special kind of covariance function `gpcf_noiset`. It is very similar to the `gpcf_noise` covariance function in that it returns diagonal covariance matrix $\mathrm{diag}(\alpha\mathbf{U})$. The scale mixture model is efficient to handle with Gibbs sampling since we are able to sample all the parameters (U_i, α, τ) efficiently from their full conditionals with regular built in samplers. For the degrees of freedom ν we use slice sampling. All the sampling steps are stored in the `gpcf_noiset` structure and `gp_mc` sampling function knows to use them if we add `gibbs_opt` field in its options structure.

Below we show the lines needed to perform the MCMC for the scale mixture model. The third line constructs the noise covariance function and the fourth line sets the option `fix_nu` to zero, which means that we are going to sample also the degrees of freedom. The degrees of freedom is many times poorly identifiable for which reason fixing its value to, for example, four is reasonable. This is the reason why it is fixed by default. With this data set its sampling is safe though. The next line initializes the GP structure and the lines after that set the sampling options. The structure `gibbs_opt` contains the options for the slice sampling used for ν. The parameters of the squared exponential covariance function are sampled with HMC and the options for this sampler are set into the structure `hmc_opt`.

```
gpcf1 = gpcf_sexp('lengthScale', 1, 'magnSigma2', 0.2^2);
```

```
gpcf1 = gpcf_sexp(gpcf1, 'lengthScale_prior', pl, 'magnSigma2_prior', pm);
lik = lik_gaussiansmt('ndata', n, 'sigma2', repmat(1,n,1), ...
                      'nu_prior', prior_logunif());

gp = gp_init('lik', lik, 'cf', {gpcf1}, 'jitterSigma2', 1e-9) %

hmc_opt.steps=10;
hmc_opt.stepadj=0.06;
hmc_opt.nsamples=1;
hmc2('state', sum(100*clock));
hmc_opt.persistence=1;
hmc_opt.decay=0.6;

% Sample
[r,g,opt]=gp_mc(gp, x, y, 'nsamples', 300, 'hmc_opt', hmc_opt);
```

The Student-t observation model with Laplace approximation

The Student-t observation model is implemented in `likelih_t`. This is used similarly to the observation models in the classification setting. The difference is that now the likelihood has also hyperparameters. These parameters can be optimized alongside the covariance function parameters with Laplace approximation. We just need to give them prior (which is by default log-uniform) and write in the parameter string, which defines the optimized parameters, 'likelihood'. All this is done with the following lines.

```
pl = prior_t();
pm = prior_t();
gpcf1 = gpcf_sexp('lengthScale', 1, 'magnSigma2', 0.2^2);
gpcf1 = gpcf_sexp(gpcf1, 'lengthScale_prior', pl, 'magnSigma2_prior', pm);

% Create the likelihood structure
pn = prior_logunif();
lik = lik_t('nu', 4, 'nu_prior', prior_logunif(), ...
            'sigma2', 10, 'sigma2_prior', pn);

% Finally create the GP data structure
gp = gp_set('lik', lik, 'cf', {gpcf1}, 'jitterSigma2', 1e-6, ...
            'latent_method', 'Laplace');

% Set the options for the scaled conjugate optimization
opt=optimset('TolFun',1e-4,'TolX',1e-4,'Display','iter','Maxiter',20);

% Optimize with the scaled conjugate gradient method
gp=gp_optim(gp,x,y,'opt',opt);

% Predictions to test points
[Eft, Varft] = gp_pred(gp, x, y, xt);
```

The Student-t observation model with MCMC

When using MCMC for the Student-t observation model we need to define sampling options for covariance function parameters, latent variables and likelihood parameters. After this we can run `gp_mc` and predict as before. All theses steps are shown below.

```
pl = prior_t();
pm = prior_sqrtunif();
gpcf1 = gpcf_sexp('lengthScale', 1, 'magnSigma2', 0.2^2);
gpcf1 = gpcf_sexp(gpcf1, 'lengthScale_prior', pl, 'magnSigma2_prior', pm);


% Create the likelihood structure
pn = prior_logunif();
lik = lik_t('nu', 4, 'nu_prior', [], 'sigma2', 10, 'sigma2_prior', pn);

% ... Finally create the GP data structure
gp = gp_set('lik', lik, 'cf', {gpcf}, 'jitterSigma2', 1e-4, ...
            'latent_method', 'MCMC');
```

```
f=gp_pred(gp,x,y,x);
gp=gp_set(gp,'latent_opt', struct('f',f));

% Set the parameters for MCMC...

% Covariance parameter-options
opt.hmc_opt.steps=5;
opt.hmc_opt.stepadj=0.05;
opt.hmc_opt.nsamples=1;

% Latent-options
opt.latent_opt.display=0;
opt.latent_opt.repeat = 10
opt.latent_opt.sample_latent_scale = 0.05

% Sample
[rgp,g,opt]=gp_mc(gp, x, y, 'nsamples', 400, opt);
```

7.2 Models for spatial epidemiology

Spatial epidemiology concerns both describing and understanding the spatial variation in the disease risk in geographically referenced health data. One of the main classes of spatial epidemiological studies is disease mapping, where the aim is to describe the overall disease distribution on a map and, for example, highlight areas of elevated or lowered mortality or morbidity risk (e.g. Lawson, 2001; Richardson, 2003; Elliot et al., 2001). The spatially referenced health data may be point level, appointing to continuously varying co-ordinates and showing for example home residence of diseased people. More commonly, however, the data are an areal level, referring to a finite subregion of a space, as for example, county or country and telling the counts of diseased people in the area (e.g. Banerjee et al., 2004).

In this section we will consider two disease mapping models. One that utilizes a Poisson observation model and other with a negative binomial observation model. The models follow the general approach discussed, for example, by Best et al. (2005). The data are aggregated into areas A_i with co-ordinates $\mathbf{x} = [x_1, x_2]^{\mathrm{T}}$. The mortality/morbidity in an area A_i is modeled as a Poisson or negative Binomial with mean $e_i\mu_i$, where e_i is the standardized expected number of cases in the area A_i, and the μ_i is the relative risk, which is given a Gaussian process prior.

The standardized expected number of cases e_i can be any positive real number that defines the expected mortality/morbidity count for the i'th area. Common practice is to evaluate it following the idea of the directly standardized rate (e.g. Ahmad et al., 2000), where the rate in an area is standardized according to the age distribution of the population in that area. The expected value in the area A_i is obtained by summing the products of the rate and population over the age-groups in the area

$$e_i = \sum_{r=1}^{R} \frac{Y_r}{N_r} n_{ir},$$

where Y_r and N_r are the total number of deaths and people in the whole area of study in the age-group r, and n_{ir} is the number of people in the age-group r and in the area A_i. In the following demos e_i and $\mathbf{y}$ are calculated from real data that contains deaths to either alcohol related diseases or cerebral vascular diseases in Finland. The examples here are based on the works by Vanhatalo and Vehtari (2007) and Vanhatalo et al. (2010).

7.2.1 Disease mapping with Poisson likelihood: demo_spatial1

The model constructed in this section is the following:

$$\mathbf{y} \sim \prod_{i=1}^{n} \text{Poisson}(y_i|\exp(f_i)e_i) \tag{7.4}$$

$$f(\mathbf{x})|\theta \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'|\theta))? \tag{7.5}$$

$$\theta \sim \text{half-Student-}t(\nu, \sigma_t^2) \tag{7.6}$$

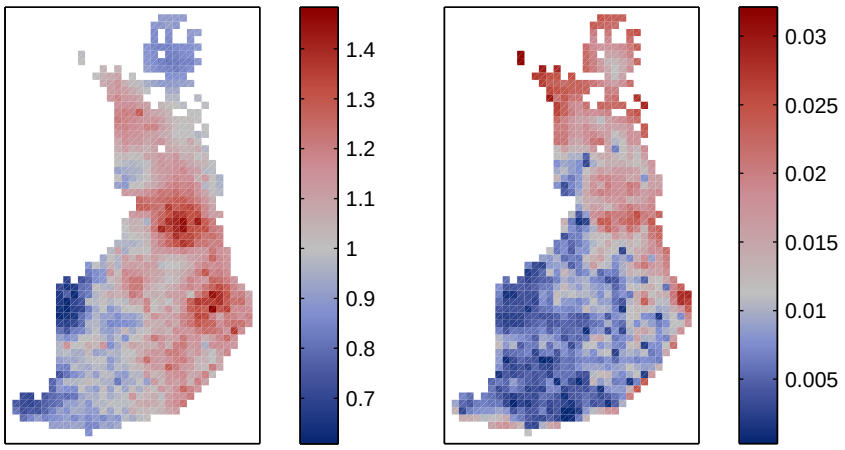
The vector $\mathbf{y}$?collects the numbers of deaths for each area. The co-ordinates of the areas are in the input vectors $\mathbf{x}$?and θ contains the covariance function parameters. The co-ordinates are defined from lower left corner of the area in 20km steps. The model is constructed with the following lines.

```
gpcf1 = gpcf_matern32('lengthScale', 5, 'magnSigma2', 0.05);
pl = prior_t('s2',10);
pm = prior_t();
gpcf1 = gpcf_matern32(gpcf1, 'lengthScale_prior', pl, 'magnSigma2_prior', pm);

lik = lik_poisson();

gp = gp_set('type', 'FIC', 'lik', lik, 'cf', {gpcf1}, [], 'X_u', Xu, ...
            'jitterSigma2', 1e-4, 'infer_params', 'covariance');

gp = gp_set(gp, 'latent_method', 'Laplace');
```

FIC sparse approximation is used since the data set is rather large and inferring full GP would be too slow. The inducing inputs `Xu` are set to a regular grid (not shown here) in the two dimensional lattice and they will be considered fixed. The extra parameter `'z'` in the last line tells the Laplace algorithm that now there is an input which affects only the likelihood. This input is stored in `ye` and it is a vector of expected number of deaths $\mathbf{e} = [e_1, ..., e_n]^{\mathrm{T}}$. In all of the previous examples we have had only inputs for the covariance function. However, if there are inputs for likelihood they should be given with optional parameter-value pair whose indicator is `'z'`. The model is now constructed and we can optimize the parameters and evaluate the posterior predictive distribution of the latent variables.

```
opt=optimset('TolFun',1e-3,'TolX',1e-3,'Display','iter');
gp=gp_optim(gp,x,y,'z',ye,'opt',opt);

[Ef, Varf] = gp_pred(gp, x, y, x, 'z', ye, 'tstind', [1:n]);
```

Here we predicted to the same locations which were used for training. Thus `Ef` and `Varf` contain the posterior mean and variance ($\mathrm{E}[\mathbf{f}|\hat{\theta}]$, $\mathrm{Var}[\mathbf{f}|\hat{\theta}]$). In this case, the prediction functions (such as `la_pred` for example) require the test index set for FIC also. This is given with parameter-value pair `'tstind', [1:n]`. These have previously been used with PIC (see section 4.2). FIC is a limiting case of PIC where each data point forms one block. Whenever we predict to new locations that have not been in the training set we do not have to worry about the test index set since all the test inputs define their own block. However, whenever we predict for exactly the same locations that are in the training set we should appoint the test inputs into the same block with the respective training input. This is done with FIC by giving `gp_pred` a vector with indices telling which of the test inputs are in the training set (`[1:n]` here). The posterior mean and variance of the latent variables are shown in the figure 7.2.



(a) The posterior mean. (b) The posterior variance.

Figure 7.2: The posterior predictive mean and variance of the latent function in the `demo_spatial1` data set obtained with FIC.

The demo contains also MCMC implementation for the model but it is not discussed here. Using Markov chain sampler for Poisson likelihood is very straightforward extension of its usage in classification model. The only difference is that we have to carry along the extra input $\mathbf{e}$.

7.2.2 Disease mapping with negative Binomial likelihood

The Negative binomial distribution is a robust version of the Poisson distribution similarly as Student-t distribution can be considered as a robustified Gaussian distribution (Gelman et al., 2004). In GPstuff it is parametrized as

$$\mathbf{y} \mid \mathbf{f}, \mathbf{e}, r \sim \prod_{i=1}^{n} \frac{\Gamma(r+y_i)}{y_i!\Gamma(r)} \left(\frac{r}{r+\mu_i}\right)^r \left(\frac{\mu_i}{r+\mu_i}\right)^{y_i} \tag{7.7}$$

$$f(\mathbf{x})|\theta \sim \mathcal{GP}\left(0, k(\mathbf{x}, \mathbf{x}'\,|\theta_g)\right), \tag{7.8}$$

$$\theta \sim \text{half-Student-}t(\nu, \sigma_t^2), \tag{7.9}$$

where $\mu_i = \mathbf{e}\exp(f(\mathbf{x}_i))$ and r is the dispersion parameter governing the variance. The model is demonstrated in `demo_spatial2` where the data are simulated so that the latent function is drawn randomly from a GP with piecewise polynomial covariance function and the observed death cases are sampled from a Negative binomial distribution. This is done in order to demonstrate the use of CS covariance functions with non-Gaussian observation model. The CS covariance functions are used just as globally supported covariance functions but are much faster. The inference in the demo is conducted with Laplace approximation and EP. The code for Laplace approximation looks the following

```
gpcf1 = gpcf_ppcs2('nin', 2, 'lengthScale', 5, 'magnSigma2', 0.05);
pl = prior_t();
pm = prior_sqrtt('s2', 0.3);
gpcf1 = gpcf_ppcs2(gpcf1, 'lengthScale_prior', pl, 'magnSigma2_prior', pm);
```

```
% Create the likelihood structure
lik = lik_negbin();

% Create the GP data structure
gp = gp_set('lik', lik, 'cf', {gpcf1}, 'jitterSigma2', 1e-4);

% Set the approximate inference method to Laplace
gp = gp_set(gp, 'latent_method', 'Laplace');

% Set the options for the scaled conjugate optimization
opt=optimset('TolFun',1e-2,'TolX',1e-2,'Display','iter');

% Optimize with the scaled conjugate gradient method
gp=gp_optim(gp,x,y,'z',ye,'opt',opt);

C = gp_trcov(gp,xx);
nnz(C) / prod(size(C))
p = amd(C);
figure
spy(C(p,p))

% make prediction to the data points
[Ef, Varf] = gp_pred(gp, x, y, x, 'z', ye);
```

7.3 Log-Gaussian Cox process

Log-Gaussian Cox-process is an inhomogeneous Poisson process model used for point data, with unknown intensity function $\lambda(\mathbf{x})$, modeled with log-Gaussian process so that $f(\mathbf{x}) = \log \lambda(\mathbf{x})$ (see Rathbun and Cressie, 1994; Møller et al., 1998). If the data are points $\mathbf{X} = \mathbf{x}_i; i = 1, 2, \ldots, n$ on a finite region $\mathcal{V}$ in $\Re^d$, then the likelihood of the unknown function f is

$$L(\mathbf{X}|f) = \exp\left\{-\left(\int_{\mathcal{V}} \exp(f(\mathbf{x}))d\,\mathbf{x}\right) + \sum_{i=1}^{n} f(\mathbf{x}_i)\right\}. \tag{7.10}$$

Evaluation of the likelihood would require nontrivial integration over the exponential of the Gaussian process. Møller et al. (1998) propose to discretise the region $\mathcal{V}$ and assume locally constant intensity in subregions. This transforms the problem to a form equivalent to having Poisson model for each subregion. Likelihood after the discretisation is

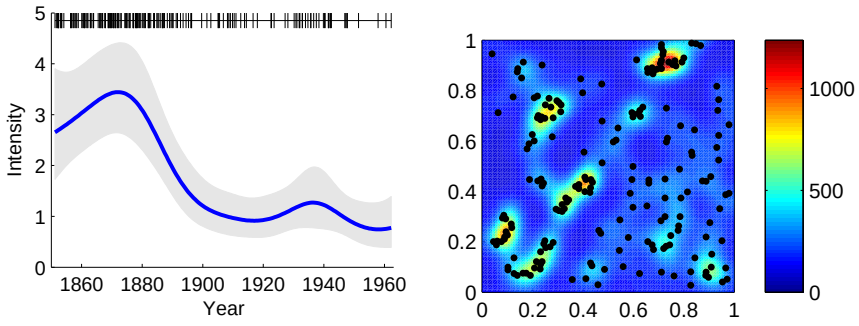
$$L(\mathbf{X}|f) \approx \prod_{k=1}^{K} \text{Poisson}(y_k|\exp(f(\dot{\mathbf{x}}_k))), \tag{7.11}$$

where $\dot{\mathbf{x}}$ is the coordinate of the kth sub-region and y_k is the number of data points in it Tokdar and Ghosh (2007) proved the posterior consistency in limit when sizes of subregions go to zero.

1D case data are the coal mine disaster data from R distribution (coal.rda) contains the dates of 191 coal mine explosions that killed ten or more men in Britain between 15 March 1851 and 22 March 1962. Computation time with expectation propagation and CCD integration over the parameters took 20s.

2D case data are the redwood data from R distribution (redwoodfull.rda) contains 195 locations of redwood trees. Computation time with Laplace approximation and MAP II for parameters took 3s.

In section 7.2.1, we demonstrated how fast no-MCMC inference for this model can be made using Laplace method or expectation propagation for integrating over the latent variables in application from spatial epidemiology. The Log-Gaussian Cox



(a) Coal mine disasters. (b) Red wood data.

Figure 7.3: Two intensity surfaces estimated with Log-Gaussian Cox process. The figures are from the `demo_lgcp`, where the aim is to study an underlying intensity surface of a point process. On the left a temporal and on the right a spatial point process.

process with the same techniques is implemented in the function `lgcp` for one or two dimensional input data. The usage of the function is demonstrated in `demo_lgcp`. This demo analyzes two data sets. The first one is one dimensional case data with coal mine disasters (from R distribution). The data contain the dates of 191 coal mine explosions that killed ten or more men in Britain between 15 March 1851 and 22 March 1962. The analysis is conducted using expectation propagation and CCD integration over the parameters and the results are shown in Figure 7.3. The second data are the redwood data (from R distribution). This data contain 195 locations of redwood trees in two dimensional lattice. The smoothed intensity surface is shown in Figure 7.3.

7.4 Binomial observation model

In this demo (`demo_binomial1`) we show how binomial likelihood is used in the GPstuff toolbox. The inference is done in this example with Laplace approximation and squared exponential covariance function.

The binomial likelihood is defined as follows:

$$p(\mathbf{y} \mid \mathbf{f}, \mathbf{z}) = \prod_{i=1}^{N} \frac{z_i!}{y_i!(z_i - y_i)!} p_i^{y_i} (1 - p_i)^{(z_i - y_i)} \tag{7.12}$$

where $p_i = \exp(f_i)/(1 + \exp(f_i))$ is the success of probability, and the vector $\mathbf{z}$ denotes the number of trials. In this demo, a Gaussian process prior is assumed for the latent variables $\mathbf{f}$.

The binomial likelihood is initialised in GPstuff as

```
% Create the likelihood structure
lik = lik_binomial();

% Create the GP data structure
gp = gp_set('lik', lik, 'cf', {gpcf1}, 'jitterSigma2', 1e-8);
```

To use binomial model, an extra parameter (the number of trials) is needed to be set as a parameter for each function that requires the data $\mathbf{y}$. For example, the model is initialized and optimized as

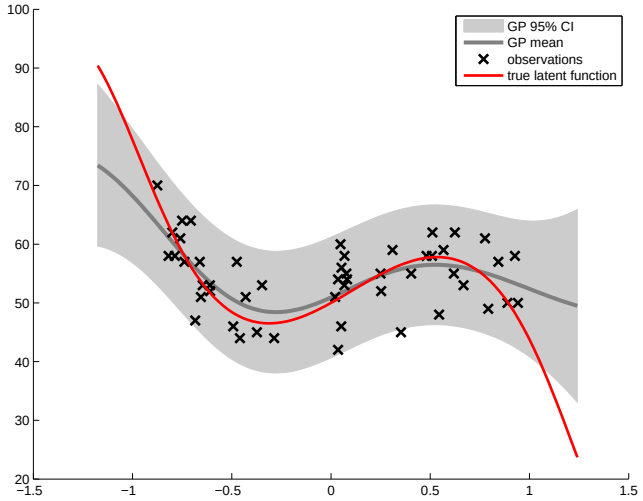


Figure 7.4: GP solutions (a MAP estimate) with squared exponential covariance function and binomial likelihood in a toy example.

```
% Set the approximate inference method
gp = gp_set(gp, 'latent_method', 'Laplace');

gp=gp_optim(gp,x,y,'z',N,'opt',opt);
```

To make predictions with binomial likelihood model without computing the predictive density, the total number of trials N_t in test points needs to be provided (in addition to N that is total number of trials in training points). In GPstuff, this is done as following:

```
% Set the total number of trials Nt at the grid points xgrid
[Eft_la, Varft_la, Eyt_la, Varyt_la] = ...
    gp_pred(gp, x, y, xgrid, 'z', N, 'zt', Ntgrid);
```

To compute the predictive densities with binomial likelihood...

```
% To compute predictive densities at the test points xt, the total number
% of trials Nt must be set additionally:
[Eft_la, Varft_la, Eyt_la, Varyt_la, pyt_la] = gp_pred(gp, x, y, xt, 'z',
N, 'yt', yt, 'zt', Nt);
```

7.5 Derivative observations in GP regression

Incorporating derivative observations in GP regression is fairly straightforward, because a derivative of Gaussian process is a Gaussian process. In short, derivative observation are taken into account by extending covariance matrices to include derivative observations. This is done by forming joint covariance matrices of function values and derivatives. Following equations (Rasmussen and Williams, 2006) state how the covariances between function values and derivatives, and between derivatives are calculated

$$\mathrm{Cov}(f_i, \frac{\partial f_j}{\partial x_{dj}}) = \frac{\partial k(\mathbf{x}_i, \mathbf{x}_j}{\partial x_{dj}}), \qquad \mathrm{Cov}(\frac{\partial f_i}{\partial x_{di}}, \frac{\partial f_j}{\partial x_{ej}}) = \frac{\partial^2 k(\mathbf{x}_i, \mathbf{x}_j}{\partial x_{di} \partial x_{ej}}.$$

The joint covariance matrix for function values and derivatives is of the following form

$$\begin{aligned}
\mathbf{K} &= \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fD} \\ \mathbf{K}_{Df} & \mathbf{K}_{DD} \end{bmatrix} \\
\mathbf{K}_{ff}^{ij} &= k(\mathbf{x}_i, \mathbf{x}_j), \\
\mathbf{K}_{Df}^{ij} &= \frac{\partial k(\mathbf{x}_i, \mathbf{x}_j)}{\partial x_{di}}, \\
\mathbf{K}_{fD} &= (\mathbf{K}_{Df})^\top, \\
\mathbf{K}_{DD}^{ij} &= \frac{\partial^2 k(\mathbf{x}_i, \mathbf{x}_j)}{\partial x_{di} \partial x_{ej}},
\end{aligned}$$

Prediction is done as usual but with derivative observations joint covariance matrices are to be used instead of the normal ones.

Using derivative observations in GPstuff requires two steps: when initializing the GP structure one must set option `'derivobs'` to `'on'`. The second step is to form right sized observation vector. With input size $n \times m$ the observation vector with derivatives should be of size $n + m \cdot n$. The observation vector is constructed by adding partial derivative observations after function value observations

$$\mathbf{y}_{obs} = \begin{bmatrix} y(\mathbf{x}) \\ \frac{\partial y(\mathbf{x})}{\partial x_1} \\ \vdots \\ \frac{\partial y(\mathbf{x})}{\partial x_m} \end{bmatrix}. \tag{7.13}$$

Different noise level could be assumed for function values and derivative observations but at the moment the implementation allows only same noise for all the observations.

7.5.1 GP regression with derivatives: demo_derivativeobs

In this section we will go through the demonstration `demo_derivativeobs`. This demo will present the main differences between GP regression with and without derivative observations and how to use them with GPstuff. The demo is divided in two parts: in the first part the GP regression is done without derivatives observations and in the second with them. Here we will present the lines of the second part because the first part is almost identical with just a few differences.

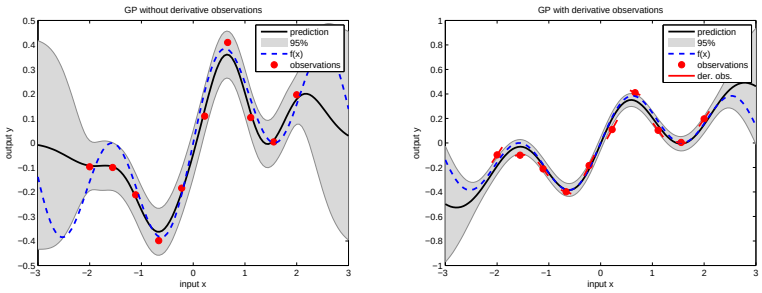
First we create the artificial data. Notice how observation vector is defined differently for GP models with and without derivatives observations. With derivative observations the observation vector includes the partial derivative observations which are set as a column vector after function value observations

```
% Create the data
tp=9;                                   %number of training points -1
x=-2:5/tp:2;
y=sin(x).*cos(x).^2;                    % The underlying process f
dy=cos(x).^3 - 2*sin(x).^2.*cos(x);     % Derivative of the process
koh=0.06;                               % noise standard deviation

% Add noise
y=y + koh*randn(size(y));
dy=dy + koh*randn(size(dy));            % derivative obs are also noisy
x=x';
dy=dy';
```

```
y=y';          % observation vector without derivative observations
y2=[y;dy];     % observation vector with derivative observations
```

The model constructed for regression is a full GP with a Gaussian likelihood. The covariance function is squared exponential, which is the only covariance function that is compatible with derivative observations at the moment. The field `derivobs` is added into `gp_set(...)` so that the inference is done with derivative observations. Field `DerivativeCheck` should also be added to `optimset(...)` when taking the predictions so that derivative observations can be used.

```
gpcf1 = gpcf_sexp('lengthScale', 0.5, 'magnSigma2', .5);
pl = prior_t();                    % a prior structure
pm = prior_sqrtt();                % a prior structure
gpcf1 = gpcf_sexp(gpcf1, 'lengthScale_prior', pl, 'magnSigma2_prior', pm);

gp = gp_set('cf', gpcf1, 'derivobs', 'on');
```



(a) Standard GP (b) GP with der. observations

Figure 7.5: GP predictions of the process f with (fig b) and without (fig a) derivative observations.

In the figure 7.5 are the predictions, observations, the underlying process and the 95 % confidence intervals for both GP model with and GP model without derivative observations.

Chapter 8

Mean functions

In the standard GP regression a zero mean function is assumed for the prior process. This is convenient but there are nonetheless some advantages in using a specified mean function. The basic principle in doing GP regression with a mean function is to apply a zero mean GP for the difference between observations and the mean function.

8.1 Explicit basis functions

Here we follow closely the presentation of (Rasmussen and Williams, 2006) about the subject and briefly present the main results. A mean function can be specified as a weighted sum of some basis functions h

$$\mathbf{m} = \mathbf{h}(\mathbf{x})^\top \boldsymbol{\beta},$$

with weights $\boldsymbol{\beta}$. The target of modeling, the underlying process $\mathbf{g}$, is assumed to be a sum of mean function and a zero mean GP.

$$\mathbf{g} = \mathbf{h}(\mathbf{x})^\top \boldsymbol{\beta} + \mathcal{GP}(0, \mathbf{K}).$$

By assuming Gaussian prior for the weights $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{b}, \mathbf{B})$, the weight parameters can be integrated out and the prior for $\mathbf{g}$ is another GP

$$\textit{prior } \mathbf{g} \sim \mathcal{GP}\left(\mathbf{h}(\mathbf{x})^\top \mathbf{b},\, \mathbf{K} + \mathbf{h}(\mathbf{x})^\top \mathbf{B}\, \mathbf{h}(\mathbf{x})\right).$$

The predictive equations are obtained by using the mean and covariance of the prior (8.1) in the zero mean GP predictive equations (2.9)

$$\mathbb{E}(\mathbf{g}_*) = \mathbb{E}(\mathbf{f}_*) + \mathbf{R}^\top \overline{\boldsymbol{\beta}}, \tag{8.1}$$

$$\mathrm{Cov}(\mathbf{g}_*) = \mathrm{Cov}(\mathbf{f}_*) + \mathbf{R}^\top \left(\mathbf{B}^{-1} + \mathbf{H}\mathbf{K}_y^{-1}\mathbf{H}^\top\right)\mathbf{R}, \tag{8.2}$$

$$\begin{aligned}
\overline{\boldsymbol{\beta}} &= \left(\mathbf{B}^{-1} + \mathbf{H}\mathbf{K}_y^{-1}\mathbf{H}^\top\right)^{-1}\left(\mathbf{B}^{-1}\mathbf{b} + \mathbf{H}\mathbf{K}_y^{-1}\mathbf{y}\right), \\
\mathbf{R} &= \mathbf{H}_* - \mathbf{H}\mathbf{K}_y^{-1}\mathbf{K}_*, \\
\mathbf{H} &= \begin{bmatrix} h_1(\mathbf{x}) \\ h_2(\mathbf{x}) \\ \vdots \\ h_k(\mathbf{x}) \end{bmatrix}, \textit{ } \boldsymbol{x} \textit{ is row vector.}
\end{aligned}$$

If the prior assumptions about the weights are vague then $\mathbf{B}^{-1} \to O$, (O is a zero matrix) and the predictive equations (8.1) and (8.2) don't depend on $\mathbf{b}$ or $\mathbf{B}$

$$\begin{aligned} \mathbb{E}(\mathbf{g}_*) &= \mathbb{E}(\mathbf{f}_*) + \mathbf{R}^\top \overline{\boldsymbol{\beta}}_{\boldsymbol{v}}, && (8.3) \\ \mathrm{Cov}(\mathbf{g}_*) &= \mathrm{Cov}(\mathbf{f}_*) + \mathbf{R}^\top \left(\mathbf{H}\mathbf{K}_y^{-1}\mathbf{H}^\top\right)\mathbf{R}, && (8.4) \end{aligned}$$

$$\overline{\boldsymbol{\beta}}_{\boldsymbol{v}} = \left(\mathbf{H}\mathbf{K}_y^{-1}\mathbf{H}^\top\right)^{-1}\mathbf{H}\mathbf{K}_y^{-1}\mathbf{y},$$

Corresponding the exact and vague prior for the basis functions' weights there are two versions of the marginal likelihood. With exact prior the marginal likelihood is

$$\begin{aligned} \log p(\mathbf{y} \mid \mathbf{X}, \mathbf{b}, \mathbf{B}) &= -\frac{1}{2}\mathbf{M}^\top\mathbf{N}^{-1}\mathbf{M} - \frac{1}{2}\log|\mathbf{K}_y| - \frac{1}{2}\log|\mathbf{B}| - \frac{1}{2}\log|\mathbf{A}| - \frac{n}{2}\log 2\pi, \\ \mathbf{M} &= \mathbf{H}^\top\mathbf{b} - \mathbf{y}, \\ \mathbf{N} &= \mathbf{K}_y + \mathbf{H}^\top\mathbf{B}\mathbf{H}, \\ \mathbf{A} &= \mathbf{B}^{-1} + \mathbf{H}\mathbf{K}_y^{-1}\mathbf{H}^\top, \end{aligned}$$

where n is the amount of observations. Its derivative with respect to hyperparameters are

$$\begin{aligned} \frac{\partial}{\partial\theta_i}\log p(\mathbf{y} \mid \mathbf{X}, \mathbf{b}, \mathbf{B}) = &+ \frac{1}{2}\mathbf{M}^\top\mathbf{N}^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_i}\mathbf{N}^{-1}\mathbf{M}^\top \\ &- \frac{1}{2}\mathrm{tr}\left(\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_i}\right) - \frac{1}{2}\mathrm{tr}\left(\mathbf{A}^{-1}\frac{\partial\mathbf{A}}{\partial\theta_i}\right), \\ \frac{\partial\mathbf{A}}{\partial\theta_i} = &- \mathbf{H}\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_i}\mathbf{K}_y^{-1}\mathbf{H}^\top. \end{aligned}$$

With a vague prior the marginal likelihood is

$$\begin{aligned} \log p(\mathbf{y} \mid \mathbf{X}) = &- \frac{1}{2}\mathbf{y}^\top\mathbf{K}_y^{-1}\mathbf{y} + \frac{1}{2}\mathbf{y}^\top\mathbf{C}\mathbf{y} \\ &- \frac{1}{2}\log|\mathbf{K}_y| - \frac{1}{2}\log|\mathbf{A}_v| - \frac{n-m}{2}\log 2\pi, \\ \mathbf{A}_v = &\ \mathbf{H}\mathbf{K}_y^{-1}\mathbf{H}^\top \\ \mathbf{C} = &\ \mathbf{K}_y^{-1}\mathbf{H}^\top\mathbf{A}_v^{-1}\mathbf{H}\mathbf{K}_y^{-1}, \end{aligned}$$

where m is the rank of $H^\top$. Its derivative is

$$\begin{aligned} \frac{\partial}{\partial\theta_i}\log p(\mathbf{y} \mid \mathbf{X}) = &\ \frac{1}{2}\mathbf{y}^\top\mathbf{P}\mathbf{y} + \frac{1}{2}\left(-\mathbf{y}^\top\mathbf{P}\mathbf{G} - \mathbf{G}^\top\mathbf{P}\mathbf{y} + \mathbf{G}^\top\mathbf{P}\mathbf{G}\right) \qquad (8.5) \\ &- \frac{1}{2}\mathrm{tr}\left(\mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_i}\right) - \frac{1}{2}\mathrm{tr}\left(\mathbf{A}_v^{-1}\frac{\partial\mathbf{A}_v}{\partial\theta_i}\right), \\ \mathbf{P} &= \mathbf{K}_y^{-1}\frac{\partial\mathbf{K}_y}{\partial\theta_i}\mathbf{K}_y^{-1}, \\ \mathbf{G} &= \mathbf{H}^\top\mathbf{A}^{-1}\mathbf{H}\mathbf{K}_y^{-1}\mathbf{y}, \end{aligned}$$

where has been used the fact that matrices $\mathbf{K}_y^{-1}$, $\frac{\partial\mathbf{K}_y}{\partial\theta_i}$ and $\mathbf{A}_v$ are symmetric. The above expression (8.5) could be simplified a little further because $\mathbf{y}^\top\mathbf{P}\mathbf{G} + \mathbf{G}^\top\mathbf{P}\mathbf{y} = 2\cdot\mathbf{y}^\top\mathbf{P}\mathbf{G}$.

8.1.1 Mean functions in GPstuff: demo_regression_meanf

Here we will demonstrate, with the `demo_regression_meanf`, how to use mean functions with GPstuff. GP regression is done for artificial one dimensional data with full GP model and gaussian likelihood. In this presentation only the lines of code are presented which are relevant in the mean function context. Otherwise this demonstration follows closely `demo_regression1`.

After creating the data and initializing the covariance and likelihood functions we are ready to initialize the GP structure. The basis functions for mean are given in a cell array of mean function structures (note the similarity to the use of covariance function structures).

```
% Create covariance and likelihood structures
gpcf = gpcf_sexp('lengthScale', [0.5], 'magnSigma2', .5);
lik = lik_gaussian('sigma2', 0.4^2);

% Initialize base functions for GP's mean function.
gpmf1 = gpmf_constant('prior_mean',.3,'prior_cov',1);
gpmf2 = gpmf_linear('prior_mean',.3,'prior_cov',1);
gpmf3 = gpmf_squared('prior_mean',.3,'prior_cov',1);

% Initialize gp structure
gp = gp_set('lik', lik, 'cf', {gpcf}, 'meanf', {gpmf1,gpmf2,gpmf3});
```

In figure 8.1 is presented the underlying process, the GP prediction, the used mean function, 95 % confidence interval and the observations.

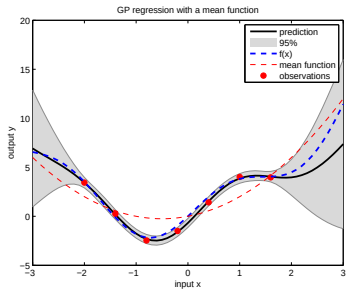


Figure 8.1: GP prediction for the process f with a mean function.

Appendix A

Function list

A.1 GP

```
THE GP TOOLS (in the GP-folder):

  Gaussian process utilities:
   GP_SET    Create and modify a Gaussian Process structure.
   GP_PAK    Combine GP parameters into one vector.
   GP_UNPAK  Set GP parameters from vector to structure
   GP_COV    Evaluate covariance matrix between two input vectors.
   GP_TRCOV  Evaluate training covariance matrix (gp_cov + noise covariance).
   GP_TRVAR  Evaluate training variance vector.
   GP_RND    Random draws from the postrior Gaussian process

  Covariance functions:
   GPCF_CAT           Create a categorigal covariance function
   GPCF_CONSTANT      Create a constant covariance function
   GPCF_EXP           Create a squared exponential covariance function
   GPCF_LINEAR        Create a linear covariance function
   GPCF_MATERN32      Create a Matern nu=3/2 covariance function
   GPCF_MATERN52      Create a Matern nu=5/2 covariance function
   GPCF_NEURALNETWORK Create a neural network covariance function
   GPCF_PERIODIC      Create a periodic covariance function
   GPCF_PPCS0         Create a piece wise polynomial (q=0) covariance function
   GPCF_PPCS1         Create a piece wise polynomial (q=1) covariance function
   GPCF_PPCS2         Create a piece wise polynomial (q=2) covariance function
   GPCF_PPCS3         Create a piece wise polynomial (q=3) covariance function
   GPCF_PROD          Create a product form covariance function
   GPCF_RQ            Create a rational quadratic covariance function
   GPCF_SEXP          Create a squared exponential covariance function

  Likelihood functions:
   LIK_GAUSSIAN    Create a Gaussian likelihood structure
   LIK_GAUSSIANSMT Create a Gaussian scale mixture approximating t
   LIK_BINOMIAL    Create a binomial likelihood structure
   LIK_LOGIT       Create a Logit likelihood structure
   LIK_NEGBIN      Create a Negbin likelihood structure
   LIK_POISSON     Create a Poisson likelihood structure
   LIK_PROBIT      Create a Probit likelihood structure
   LIK_T           Create a Student-t likelihood structure

 Inference utilities:
   GP_E          Evaluate energy function (un-normalized negative marginal
                 log posterior)
   GP_G          Evaluate gradient of energy (GP_E) for Gaussian Process
   GP_EG         Evaluate both GP_E and GP_G. Useful in optimisation.
   GP_PRED       Make predictions with Gaussian process
   GPEP_E        Conduct Expectation propagation and return negative marginal
                 log posterior estimate
   GPEP_G        Evaluate gradient of EP's negative marginal log posterior
                 estimate
   GPEP_PRED     Predictions with Gaussian Process EP approximation
```

```
 GPLA_E          Construct Laplace approximation and return negative marginal
                 log posterior estimate
 GPLA_G          Evaluate gradient of Laplace approximation's marginal
                 log posterior estimate
 GPLA_PRED       Predictions with Gaussian Process Laplace approximation
 GP_MC           Markov chain sampling for Gaussian process models
 GPMC_PRED       Predictions with Gaussian Process MCMC approximation.
 GPMC_PREDS      Conditional predictions with Gaussian Process MCMC
                 approximation.
 GP_IA           Integration approximation with grid, Monte Carlo or
                 CCD integration
 GPIA_PRED       Prediction with Gaussian Process GP_IA solution.
 LGCP            Log Gaussian Cox Process intensity estimate for 1D and
                 2D data

Model assesment and comparison:
 GP_DIC          The DIC statistics and efective number of parameters in a GP model
 GP_KFCV         K-fold cross validation for a GP model
 GP_LOOE         Evaluate the leave-one-out predictive density in case of
                 Gaussian observation model
 GP_LOOG         Evaluate the gradient of the leave-one-out predictive
                 density (GP_LOOE) in case of Gaussian observation model
 GP_EPLOOE       Evaluate the leave-one-out predictive density in case of
                 non-Gaussian observation model and EP
 EP_LOOPRED      Leave-one-out-predictions with Gaussian Process EP approximation
 GP_PEFF         The efective number of parameters in GP model with focus
                 on latent variables.

Metrics:
 METRIC_EUCLIDEAN   An Euclidean distance for Gaussian process models.

Misc:
   LDLROWMODIFY  Function to modify the sparse cholesky factorization
                 L*D*L' = C, when a row and column k of C have changed
   LDLROWUPDATE  Multiple-rank update or downdate of a sparse LDL' factorization.
   SPINV         Evaluate the sparsified inverse matrix
   SCALED_HMC    A scaled hybric Monte Carlo samping for latent values
   SCALED_MH     A scaled Metropolis Hastings samping for latent values
   GP_INSTALL    Matlab function to compile all the c-files to mex in the
                 GPstuff/gp folder.

Demonstration programs:
 DEMO_BINOMIAL1          Demonstration of Gaussian process model with binomial
                         likelihood
 DEMO_BINOMIAL_APC       Demonstration for modeling age-period-cohort data
                         by a binomial model combined with GP prior.
 DEMO_CLASSIFIC          Classification problem demonstration for 2 classes
 DEMO_LGCP               Demonstration for a log Gaussian Cox process
                         with inference via EP or Laplace approximation
 DEMO_MODELASSESMENT1    Demonstration for model assesment with DIC, number
                         of effective parameters and ten-fold cross validation
 DEMO_MODELASSESMENT2    Demonstration for model assesment when the observation
                         model is non-Gaussian
 DEMO_NEURALNETWORKCOV   Demonstration of Gaussian process with a neural
                         network covariance function
 DEMO_PERIODIC           Regression problem demonstration for periodic data
 DEMO_REGRESSION1        Regression problem demonstration for 2-input
                         function with Gaussian process
 DEMO_REGRESSION_PPCS    Regression problem demonstration for 2-input
                         function with Gaussian process using CS covariance
 DEMO_REGRESSION_ADDITIVE1 Regression problem demonstration with additive model
 DEMO_REGRESSION_ADDITIVE2 Regression demonstration with additive Gaussian
                         process using linear, squared exponential and
                         neural network covariance fucntions
 DEMO_REGRESSION_HIER    Hierarchical regression demonstration
 DEMO_REGRESSION_ROBUST  A regression demo with Student-t distribution as a
                         residual model.
 DEMO_REGRESSION_SPARSE1 Regression problem demonstration for 2-input
                         function with sparse Gaussian processes
 DEMO_REGRESSION_SPARSE2 Regression demo comparing different sparse
                         approximations
 DEMO_SPATIAL1           Demonstration for a disease mapping problem
                         with Gaussian process prior and Poisson likelihood
 DEMO_SPATIAL2           Demonstration for a disease mapping problem with
```

```
                        Gaussian process prior and negative binomial
                        observation model
```

A.2 Diagnostic tools

```
THE DIAGNOSTIC TOOLS (in the diag-folder):

Covergence diagnostics
  PSRF     - Potential Scale Reduction Factor
  CPSRF    - Cumulative Potential Scale Reduction Factor
  MPSRF    - Multivariate Potential Scale Reduction Factor
  CMPSRF   - Cumulative Multivariate Potential Scale Reduction Factor
  IPSRF    - Interval-based Potential Scale Reduction Factor
  CIPSRF   - Cumulative Interval-based Potential Scale Reduction Factor
  KSSTAT   - Kolmogorov-Smirnov goodness-of-fit hypothesis test
  HAIR     - Brooks' hairiness convergence diagnostic
  CUSUM    - Yu-Mykland convergence diagnostic for MCMC
  SCORE    - Calculate score-function convergence diagnostic
  GBINIT   - Initial iterations for Gibbs iteration diagnostic
  GBITER   - Estimate number of additional Gibbs iterations

Time series analysis
  ACORR      - Estimate autocorrelation function of time series
  ACORRTIME  - Estimate autocorrelation evolution of time series (simple)
  GEYER_ICSE - Compute autocorrelation time tau using Geyer's
               initial convex sequence estimator
               (requires Optimization toolbox)
  GEYER_IMSE - Compute autocorrelation time tau using Geyer's
               initial monotone sequence estimator

Kernel density estimation etc.:
  KERNEL1  - 1D Kernel density estimation of data
  KERNELS  - Kernel density estimation of independent components of data
  KERNELP  - 1D Kernel density estimation, with automatic kernel width
  NDHIST   - Normalized histogram of N-dimensional data
  HPDI     - Estimates the Bayesian HPD intervals

Manipulation of MCMC chains
  THIN     - Delete burn-in and thin MCMC-chains
  JOIN     - Join similar structures of arrays to one structure of arrays
  BATCH    - Batch MCMC sample chain and evaluate mean/median of batches

Misc:
  CUSTATS   - Calculate cumulative statistics of data
  BBPRCTILE - Bayesian bootstrap percentile
  GRADCHEK  - Checks a user-defined gradient function using finite
              differences.
  DERIVATIVECHECK - Compare user-supplied derivatives to
                    finite-differencing derivatives.
```

A.3 Distributions

```
PROBABILITY DISTRIBUTION FUNCTIONS (in the dist-folder):

Priors
 PRIOR_FIXED       Fix parameter to its current value
 PRIOR_GAMMA       Gamma prior structure
 PRIOR_INVGAMMA    Inverse-gamma prior structure
 PRIOR_LAPLACE     Laplace (double exponential) prior structure
 PRIOR_LOGLOGUNIF  Uniform prior structure for the log(log(parameter))
 PRIOR_LOGUNIF     Uniform prior structure for the logarithm of the parameter
 PRIOR_GAUSSIAN    Gaussian prior structure
 PRIOR_LOGGAUSSIAN Log-Gaussian prior structure
 PRIOR_SINVCHI2    Scaled inverse-chi-square prior structure
 PRIOR_T           Student-t prior structure
 PRIOR_SQRTT       Student-t prior structure for the square root of the
                   parameter
 PRIOR_UNIF        Uniform prior structure
 PRIOR_SQRTUNIF    Uniform prior structure for the square root of the
```

```
                  parameter

Probability density functions

   BETA_LPDF     - Beta log-probability density function (lpdf).
   BETA_PDF      - Beta probability density function (pdf).
   DIR_LPDF      - Log probability density function of uniform Dirichlet
                   distribution
   DIR_PDF       - Probability density function of uniform Dirichlet
                   distribution
   GAM_CDF       - Cumulative of Gamma probability density function (cdf).
   GAM_LPDF      - Log of Gamma probability density function (lpdf).
   GAM_PDF       - Gamma probability density function (pdf).
   GEO_LPDF      - Geometric log probability density function (lpdf).
   INVGAM_LPDF   - Inverse-Gamma log probability density function.
   INVGAM_PDF    - Inverse-Gamma probability density function.
   LAPLACE_LPDF  - Laplace log-probability density function (lpdf).
   LAPLACE_PDF   - Laplace probability density function (pdf).
   LOGN_LPDF     - Log normal log-probability density function (lpdf)
   LOGT_LPDF     - Log probability density function (lpdf) for log Student's T
   MNORM_LPDF    - Multivariate-Normal log-probability density function (lpdf).
   MNORM_PDF     - Multivariate-Normal log-probability density function (lpdf).
   NORM_LPDF     - Normal log-probability density function (lpdf).
   NORM_PDF      - Normal probability density function (pdf).
   POISS_LPDF    - Poisson log-probability density function.
   POISS_PDF     - Poisson probability density function.
   SINVCHI2_LPDF - Scaled inverse-chi log-probability density function.
   SINVCHI2_PDF  - Scaled inverse-chi probability density function.
   T_LPDF        - Student's T log-probability density function (lpdf)
   T_PDF         - Student's T probability density function (pdf)

Random number generators

   CATRAND       - Random matrices from categorical distribution.
   DIRRAND       - Uniform dirichlet random vectors
   EXPRAND       - Random matrices from exponential distribution.
   GAMRAND       - Random matrices from gamma distribution.
   INTRAND       - Random matrices from uniform integer distribution.
   INVGAMRAND    - Random matrices from inverse gamma distribution
   INVGAMRAND1   - Random matrices from inverse gamma distribution
   INVWISHRND    - Random matrices from inverse Wishart distribution.
   NORMLTRAND    - Random draws from a left-truncated normal
                   distribution, with mean = mu, variance = sigma2
   NORMRTRAND    - Random draws from a right-truncated normal
                   distribution, with mean = mu, variance = sigma2
   NORMTRAND     - Random draws from a normal truncated to interval
   NORMTZRAND    - Random draws from a normal distribution truncated by zero
   SINVCHI2RAND  - Random matrices from scaled inverse-chi distribution
   TRAND         - Random numbers from Student's t-distribution
   UNIFRAND      - Generate unifrom random numberm from interval [A,B]
   WISHRND       - Random matrices from Wishart distribution.

Others
   KERNELP       - Kernel density estimator for one dimensional distribution.
   HAMMERSLEY    - Hammersley quasi-random sequence
```

A.4 Monte Carlo

```
MONTE CARLO FUNCTIONS (in the mc-folder):

   BBMEAN        - Bayesian bootstrap mean
   GIBBS         - Gibbs sampling
   HMC2          - Hybrid Monte Carlo sampling.
   HMC2_OPT      - Default options for Hybrid Monte Carlo sampling.
   HMEAN         - Harmonic mean
   METROP2       - Markov Chain Monte Carlo sampling with Metropolis algorithm.
   METROP2_OPT   - Default options for Metropolis sampling.
   RESAMPDET     - Deterministic resampling
   RESAMPRES     - Residual resampling
   RESAMPSIM     - Simple random resampling
   RESAMPSTR     - Stratified resampling
   SLS           - Markov Chain Monte Carlo sampling using Slice Sampling
```

```
   SLS_OPT       - Default options for Slice Sampling
   SLS1MM        - 1-dimensional fast minmax Slice Sampling
   SLS1MM_OPT    - Default options for SLS1MM_OPT
   SOFTMAX2      - Softmax transfer function
```

A.5 Miscellaneous

```
MISCELLANEOUS FUNCTIONS (in the misc-folder):)

   CVIT         - Create itr and itst indeces for k-fold-cv
   CVITR        - Create itr and itst indeces for k-fold-cv with ranndom
                  permutation
   MAPCOLOR     - returns a colormap ranging from blue through gray
                  to red
   MAPCOLOR2    - Create a blue-gray-red colormap.
   M2KML        - Converts GP prediction results to a KML file
   QUAD_MOMENTS - Calculate the 0th, 1st and 2nd moment of a given (unnormalized)
                  probability distribution
   RANDPICK     - Pick element from x randomly
                  If x is matrix, pick row from x randomly.
   STR2FUN      - Compatibility wrapper to str2func
   SET_PIC      - Set the inducing inputs and blocks for two dimensional input data
   WMEAN        - weighted mean
```

A.6 Optimization

```
OPTIMIZATION FUNCTIONS (in the optim-folder):

   BSEARCH       - Finds the minimum of a combinatorial function using backward search
   BSEARCH_OPT   - Default options for backward search
   FSEARCH       - Finds the minimum of a combinatorial function using forward search
   FSEARCH_OPT   - Default options for forward search
   SCGES         - Scaled conjugate gradient optimization with early stopping
   SCGES_OPT     - Default options for scaled conjugate gradient optimization
   SCGES         - Scaled conjugate gradient optimization with early stopping (new options structure).
   SCG2          - Scaled conjugate gradient optimization
   SCG2_OPT      - Default options for scaled conjugate gradient optimization (scg2) (new options structure).
   FMINSCG       - Scaled conjugate gradient optimization
   FMINLBFGS     - (Limited memory) Quasi Newton
                   Broyden-Fletcher-Goldfarb-Shanno (BFGS)
```

Appendix B

Covariance functions

In this section we summarize all the covariance functions in the GPstuff package.

Squared exponential covariance function (`gpcf_sexp`)

Probably the most widely-used covariance function is the squared exponential (SE)

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2_{\text{sexp}} \exp\left(-\frac{1}{2} \sum_{k=1}^{d} \frac{(x_{i,k} - x_{j,k})^2}{l_k^2} \right). \tag{B.1}$$

The length scale l_k governs the correlation scale in input dimension k and the magnitude σ^2_{sexp} the overall variability of the process. A squared exponential covariance function leads to very smooth GPs that are infinitely times mean square differentiable.

Exponential covariance function (`gpcf_exp`)

Exponential covariance function is defined as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2_{\text{exp}} \exp\left(-\sqrt{\sum_{k=1}^{d} \frac{(x_{i,k} - x_{j,k})^2}{l_k^2}} \right). \tag{B.2}$$

The parameters l_k and σ^2_{exp} have similar role as with the SE covariance function. The exponential covariance function leads to very rough GPs that are not mean square differentiable.

M?ern class of covariance functions (`gpcf_maternXX`)

The Matern class of covariance functions is given by

$$k_\nu(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2_{\text{m}} \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} r \right)^\nu K_\nu \left(\sqrt{2\nu} r \right), \tag{B.3}$$

where $r = \left(\sum_{k=1}^{d} \frac{(x_{i,k} - x_{j,k})^2}{l_k^2} \right)^{1/2}$. The parameter ν governs the smoothness of the process, and K_ν is a modified Bessel function (Abramowitz and Stegun, 1970, sec. 9.6). The Matern covariance functions can be represent in a simpler form when ν is a

half integer. The Mátern covariance functions with $\nu = 3/2$ (`gpcf_matern32`) and $\nu = 5/2$ (`gpcf_matern52`) are:

$$k_{\nu=3/2}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_{\mathrm{m}}^2 \left(1 + \sqrt{3}r\right) \exp\left(-\sqrt{3}r\right) \tag{B.4}$$

$$k_{\nu=5/2}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_{\mathrm{m}}^2 \left(1 + \sqrt{5}r + \frac{5r^2}{3}\right) \exp\left(-\sqrt{5}r\right). \tag{B.5}$$

Neural network covariance function (`gpcf_neuralnetwork`)

A neural network with suitable transfer function and prior distribution converges to a GP as the number of hidden units in the network approaches to infinity (Neal, 1996; Williams, 1996; Rasmussen and Williams, 2006). A nonstationary neural network covariance function is

$$k(\mathbf{x}_i, \mathbf{x}_j) = \frac{2}{\pi} \sin^{-1}\left(\frac{2\tilde{\mathbf{x}}_i^{\mathrm{T}} \Sigma \tilde{\mathbf{x}}_j}{(1 + 2\tilde{\mathbf{x}}_i^{\mathrm{T}} \Sigma \tilde{\mathbf{x}}_i)(1 + 2\tilde{\mathbf{x}}_j^{\mathrm{T}} \Sigma \tilde{\mathbf{x}}_j)}\right), \tag{B.6}$$

where $\tilde{\mathbf{x}} = (1, x_1, \ldots, x_d)^{\mathrm{T}}$ is an input vector augmented with 1. $\Sigma = \mathrm{diag}(\sigma_0^2, \sigma_1^2, \ldots, \sigma_d^2)$ is a diagonal weight prior, where σ_0^2 is a variance for bias parameter controlling the functions offset from the origin. The variances for weight parameters are $\sigma_1^2, \ldots, \sigma_d^2$, and with small values for weights, the neural network covariance function produces smooth and rigid looking functions. The larger values for the weight variances produces more flexible solutions.

Constant covariance function (`gpcf_constant`)

Perhaps the simplest covariance function is the constant covariance function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \tag{B.7}$$

with variance parameter σ^2. This function can be used to implement the constant term in the dotproduct covariance function (Rasmussen and Williams, 2006) reviewed below.

Linear covariance function (`gpcf_linear`)

The linear covariance function is

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^{\mathrm{T}} \Sigma \mathbf{x}_j \tag{B.8}$$

where the diagonal matrix $\Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_D^2)$ contains the prior variances of the linear model coefficients. Combining this with the constant function above we can form covariance function Rasmussen and Williams (2006), which calls a dotproduct covariance function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 + \mathbf{x}_i^{\mathrm{T}} \Sigma \mathbf{x}_j. \tag{B.9}$$

Piecewise polynomial functions (`gpcf_ppcsX`)

The piecewise polynomial functions are the only compactly supported covariance functions (see section 4) in GPstuff. There are four of them with the following forms

$$k_{pp0}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2(1-r)_+^j \tag{B.10}$$

$$k_{pp1}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2(1-r)_+^{j+1}\left((j+1)r+1\right) \tag{B.11}$$

$$k_{pp2}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sigma^2}{3}(1-r)_+^{j+2}((j^2+4j+3)r^2+(3j+6)r+3) \tag{B.12}$$

$$\begin{aligned} k_{pp3}(\mathbf{x}_i, \mathbf{x}_j) =& \frac{\sigma^2}{15}(1-r)_+^{j+3}((j^3+9j^2+23j+15)r^3+ \\ & (6j^2+36j+45)r^2+(15j+45)r+15) \end{aligned} \tag{B.13}$$

where $j = \lfloor d/2 \rfloor + q + 1$. These functions correspond to processes that are $2q$ times mean square differentiable at the zero and positive definite up to the dimension d (Wendland, 2005). The covariance functions are named `gpcf_ppcs0`, `gpcf_ppcs1`, `gpcf_ppcs2`, and `gpcf_ppcs3`.

Rational quadratic covariance function (`gpcf_rq`)

The rational quadratic (RQ) covariance function (Rasmussen and Williams, 2006)

$$k_{\mathrm{RQ}}(\mathbf{x}_i, \mathbf{x}_j) = \left(1 + \frac{1}{2\alpha}\sum_{k=1}^{d}\frac{(x_{i,k}-x_{j,k})^2}{l_k^2}\right)^{-\alpha} \tag{B.14}$$

can be seen as a scale mixture of squared exponential covariance functions with different length-scales. The smaller the parameter $\alpha > 0$ is the more diffusive the length-scales of the mixing components are. The parameter $l_k > 0$ characterizes the typical length-scale of the individual components in input dimension k.

Periodic covariance function (`gpcf_periodic`)

Many real world systems exhibit periodic phenomena, which can be modelled with a periodic covariance function. One possible construction (Rasmussen and Williams, 2006) is

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\sum_{k=1}^{d}\frac{2\sin^2(\pi(x_{i,k}-x_{j,k})/\gamma)}{l_k^2}\right), \tag{B.15}$$

where the parameter γ controls the inverse length of the periodicity and l_k the smoothness of the process in dimension k.

Product covariance function (`gpcf_product`)

A product of two or more covariance functions, $k_1(\mathbf{x}, \mathbf{x}') \cdot k_2(\mathbf{x}, \mathbf{x}')...$, is a valid covariance function as well. Combining covariance functions in a product form can be done with `gpcf_prod`, for which the user can freely specify the covariance functions to be multiplied with each other from the collection of covariance functions implemented in GPstuff.

Categorical covariance function (`gpcf_cat`)

Categorical covariance function `gpcf_cat` returns correlation 1 if input values are equal and 0 otherwise.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 1 & \text{if } \mathbf{x}_i - \mathbf{x}_j = 0 \\ 0 & \text{otherwise} \end{cases} \tag{B.16}$$

Categorical covariance function can be combined with other covariance functions using `gpcf_prod`, for example, to produce hierarchical models.

Appendix C

Observation models

Here, we summarize all the observation models in GPstuff. Most of them are implemented in files `lik_*` which reminds that at the inference step they are considered likelihood functions.

Gaussian (`lik_gaussian`)

The i.i.d Gaussian noise with variance σ^2 is

$$\mathbf{y} \,|\, \mathbf{f}, \sigma^2 \sim N(\mathbf{f}, \sigma^2 \mathbf{I}). \tag{C.1}$$

Student-t (`lik_t`, `lik_gaussiansmt`)

The Student-t observation model (implemented in `lik_t`) is

$$\mathbf{y} \,|\, \mathbf{f}, \nu, \sigma_t \sim \prod_{i=1}^{n} \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)\sqrt{\nu\pi}\sigma_t} \left(1 + \frac{(y_i - f_i)^2}{\nu \sigma_t^2}\right)^{-(\nu+1)/2}, \tag{C.2}$$

where ν is the degrees of freedom and σ the scale parameter. The scale mixture version of the Student-t distribution is implemented in `lik_gaussiansmt` and it is parametrized as

$$y_i | f_i, \alpha, U_i \sim N(f_i, \alpha U_i) \tag{C.3}$$

$$U_i \sim \text{Inv-}\chi^2(\nu, \tau^2), \tag{C.4}$$

where each observation has its own noise variance αU_i (Neal, 1997; Gelman et al., 2004).

Logit (`lik_logit`)

The logit transformation gives the probability for y_i of being 1 or -1 as

$$p_{\text{logit}}(y_i | f_i) = \frac{1}{1 + \exp(-y_i f_i)}. \tag{C.5}$$

Probit (`lik_probit`)

The probit transformation gives the probability for y_i of being 1 or -1 as

$$p_{\text{probit}}(y_i|f_i) = \Phi(y_i f(\mathbf{x}_i)) = \int_{-\infty}^{y_i f(\mathbf{x}_i)} N(z|0,1)dz. \tag{C.6}$$

Poisson (`lik_poisson`)

The Poisson observation model with expected number of cases $\mathbf{e}$ is

$$\mathbf{y} \,|\, \mathbf{f}, \mathbf{e} \sim \prod_{i=1}^{n} \text{Poisson}(y_i|\exp(f_i)e_i). \tag{C.7}$$

Negative-Binomial (`lik_negbin`)

The negative-binomial is a robustified version of the Poisson distribution. It is parametrized

$$\mathbf{y} \,|\, \mathbf{f}, \mathbf{e}, r \sim \prod_{i=1}^{n} \frac{\Gamma(r+y_i)}{y_i!\Gamma(r)} \left(\frac{r}{r+\mu_i}\right)^r \left(\frac{\mu_i}{r+\mu_i}\right)^{y_i} \tag{C.8}$$

where $\mu_i = \mathbf{e}\exp(f_i)$, r is the dispersion parameter governing the variance, e_i is the expected number of cases and y_i is positive integer telling the observed count.

Binomial (`lik_binomial`)

The binomial observation model with the probability of success $p_i = \exp(f_i)/(1+\exp(f_i))$ is

$$\mathbf{y} \,|\, \mathbf{f}, \mathbf{z} \sim \prod_{i=1}^{N} \frac{z_i!}{y_i!(z_i-y_i)!} p_i^{y_i}(1-p_i)^{(z_i-y_i)}. \tag{C.9}$$

Here, z_i denotes the number of trials and y_i is the number of successes.

Appendix D

Priors

This chapter lists all the priors implemented in the GPstuff package.

Gaussian prior (`prior_gaussian`)

The Gaussian distribution is parametrized as

$$p(\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(\theta-\mu)^2\right) \tag{D.1}$$

where μ is a location parameter and σ^2 is a scale parameter.

Log-Gaussian prior (`prior_loggaussian`)

The log-Gaussian distribution is parametrized as

$$p(\theta) = \frac{1}{\theta\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(\log(\theta)-\mu)^2\right) \tag{D.2}$$

where μ is a location parameter and σ^2 is a scale parameter.

Laplace prior (`prior_laplace`)

The Laplace distribution is parametrized as

$$p(\theta) = \frac{1}{2\sigma} \exp\left(-\frac{|\theta-\mu|}{\sigma}\right) \tag{D.3}$$

where μ is a location parameter and $\sigma > 0$ is a scale parameter.

Student-t prior (`prior_t`)

The Student-t distribution is parametrized as

$$p(\theta) = \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)\sqrt{\nu\pi\sigma^2}} \left(1 + \frac{(\theta-\mu)^2}{\nu\sigma^2}\right)^{-(\nu+1)/2} \tag{D.4}$$

where μ is a location parameter, σ^2 is a scale parameter and $\nu > 0$ is the degrees of freedom.

Square root Student-t prior (`prior_sqrtt`)

The square root Student-t distribution is parametrized as

$$p(\theta^{1/2}) = \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)\sqrt{\nu\pi\sigma^2}} \left(1 + \frac{(\theta-\mu)^2}{\nu\sigma^2}\right)^{-(\nu+1)/2} \tag{D.5}$$

where μ is a location parameter, σ^2 is a scale parameter and $\nu > 0$ is the degrees of freedom.

Scaled inverse-χ^2 prior (`prior_sinvchi2`)

The scaled inverse-χ^2 distribution is parametrized as

$$p(\theta) = \frac{(\nu/2)^{\nu/2}}{\Gamma(\nu/2)} (s^2)^{\nu/2} \theta^{-(\nu/2+1)} e^{-\nu s^2/(2\theta)} \tag{D.6}$$

where s^2 is a scale parameter and $\nu > 0$ is the degrees of freedom parameter.

Gamma prior (`prior_gamma`)

The gamma distribution is parametrized as

$$p(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \theta^{\alpha-1} e^{-\beta\theta} \tag{D.7}$$

where $\alpha > 0$ is a shape parameter and $\beta > 0$ is an inverse scale parameter.

Inverse-gamma prior (`prior_invgamma`)

The inverse-gamma distribution is parametrized as

$$p(\theta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \theta^{-(\alpha+1)} e^{-\beta/\theta} \tag{D.8}$$

where $\alpha > 0$ is a shape parameter and $\beta > 0$ is a scale parameter.

Uniform prior (`prior_unif`)

The uniform prior is parametrized as

$$p(\theta) \propto 1. \tag{D.9}$$

Square root uniform prior (`prior_sqrtunif`)

The square root uniform prior is parametrized as

$$p(\theta^{1/2}) \propto 1. \tag{D.10}$$

Log-uniform prior (`prior_logunif`)

The log-uniform prior is parametrized as

$$p(\log(\theta)) \propto 1. \tag{D.11}$$

Log-log-uniform prior (`prior_loglogunif`)

The log-log-uniform prior is parametrized as

$$p(\log(\log(\theta))) \propto 1. \tag{D.12}$$

Appendix E

Transformation of hyperparameters

The inference on the parameters of covariance functions is conducted mainly transformed space. Most of ten used transformation is log-transformation, which has the advantage that the parameter space $(0, \infty)$ is transformed into $(-\infty, \infty)$. The change of parametrization has to be taken into account in the evaluation of the probability densities of the model. If parameter θ with probability density $p_\theta(\theta)$ is transformed into the parameter $w = f(\theta)$ with equal number of components, the probability density of w is given by

$$p_w(w) = |J| p_\theta(f^{-1}(w)), \tag{E.1}$$

where J is the Jacobian of the transformation $\theta = f^{-1}(w)$. The parameter transformations are discussed shortly, for example, in Gelman et al. (2004)[p. 24].

Due to the log transformation $w = \log(\theta)$ transformation the probability densities $p_\theta(\theta)$ are changed to the densities

$$p_w(w) = |J| p_\theta(\exp(w)) = |J| p_\theta(\theta), \tag{E.2}$$

where the Jacobian is $J = \frac{\partial \exp(w)}{\partial w} = \exp(w) = \theta$. Now, given Gaussian observation model (see Section 2.1.2) the posterior of w can be written as

$$p_w(w|\mathcal{D}) \propto p(\mathbf{y}|\mathbf{X}, \theta) p(\theta|\gamma)\theta, \tag{E.3}$$

which leads to energy function

$$\begin{aligned} E(w) &= -\log p(\mathbf{y}|\mathbf{X}, \theta) - \log p(\theta|\gamma) - \log(|\theta|). \\ &= E(\theta) - \log(\theta), \end{aligned}$$

where the absolute value signs are not shown explicitly around θ because it is strictly positive. Thus, the log transformation just adds term $-\log \theta$ in the energy function.

The inference on w requires also the gradients of an energy function $E(w)$. These

can be obtained easily with the chain rule

$$\begin{aligned}\frac{\partial E(w)}{\partial w} &= \frac{\partial}{\partial \theta}\left[E(\theta) - \log(|J|)\right]\frac{\partial \theta}{\partial w} \\ &= \left[\frac{\partial E(\theta)}{\partial \theta} - \frac{\partial \log(|J|)}{\partial \theta}\right]\frac{\partial \theta}{\partial w} \\ &= \left[\frac{\partial E(\theta)}{\partial \theta} - \frac{1}{|J|}\frac{\partial |J|}{\partial \theta}\right] J. \end{aligned} \tag{E.4}$$

Here we have used the fact that the last term, derivative of θ with respect to w, is the same as the Jacobian $J = \frac{\partial \theta}{\partial w} = \frac{\partial f^{-1}}{\partial w}$. Now in the case of log transformation the Jacobian can be replaced by θ and the gradient is gotten an easy expression

$$\frac{\partial E(w)}{\partial w} = \frac{\partial E(\theta)}{\partial \theta}\theta - 1. \tag{E.5}$$

Bibliography

Abramowitz, M. and Stegun, I. A. (1970). *Handbook of mathematical functions*. Dover Publications, Inc.

Ahmad, O. B., Boschi-Pinto, C., Lopez, A. D., Murray, C. J., Lozano, R., and Inoue, M. (2000). Age standardization of rates: A new WHO standard. *GPE Discussion Paper Series*, 31.

Alvarez, M. A., Luengo, D., Titsias, M. K., and Lawrence, N. D. (2010). Efficient multioutput Gaussian processes through variational inducing kernels. *JMLR Workshop and conference proceedings*, 9:25–32.

Banerjee, S., Carlin, B. P., and Gelfand, A. E. (2004). *Hierarchical Modelling and Analysis for Spatial Data*. Chapman Hall/CRC.

Banerjee, S., Gelfand, A. E., Finley, A. O., and Sang, H. (2008). Gaussian predictive process models for large spatial data sets. *Journal of the Royal Statistical Society B*, 70(4):825–848.

Bernardo, J. M. (1979). Expected information as expected utility. *Annals of Statistics*, 7(3):686–690.

Bernardo, J. M. and Smith, A. F. M. (2000). *Bayesian Theory*. John Wiley & Sons, Ltd.

Best, N., Richardson, S., and Thomson, A. (2005). A comparison of Bayesian spatial models for disease mapping. *Statistical Methods in Medical Research*, 14:35–59.

Buhmann, M. D. (2001). A new class of radial basis functions with compact support. *Mathematics of Computation*, 70(233):307–318.

Burman, P. (1989). A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. *Biometrika*, 76(3):503–514.

Christensen, O. F., Roberts, G. O., and Sköld, M. (2006). Robust Markov chain Monte Carlo methods for spatial generalized linear mixed models. *Journal of Computational and Graphical Statistics*, 15:1–17.

Cressie, N. A. C. (1993). *Statistics for Spatial Data*. John Wiley & Sons, Inc.

Csató, L. and Opper, M. (2002). Sparse online Gaussian processes. *Neural Computation*, 14(3):641–669.

Davis, T. A. (2006). *Direct Methods for Sparse Linear Systems*. SIAM.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924.

Diggle, P. J. and Ribeiro, P. J. (2007). *Model-based Geostatistics*. Springer Science+Business Media, LLC.

Diggle, P. J., Tawn, J. A., and Moyeed, R. A. (1998). Model-based geostatistics. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 47(3):299–350.

Duane, S., Kennedy, A., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222.

Elliot, P., Wakefield, J., Best, N., and Briggs, D., editors (2001). *Spatial Epidemiology Methods and Applications*. Oxford University Press.

Finkenstädt, B., Held, L., and Isham, V. (2007). *Statistical Methods for Spatio-Temporal Systems*. Chapman & Hall/CRC.

Furrer, R., Genton, M. G., and Nychka, D. (2006). Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, 15(3):502–523.

Gaspari, G. and Cohn, S. (1999). Construction of correlation functions in two and three dimensions. *Quarterly Journal of the Royal Meteorological Society*, 125(554):723–757.

Gelfand, A. E., Diggle, P. J., Fuentes, M., and Guttorp, P. (2010). *Handbook of Spatial Statistics*. CRC Press.

Gelman, A. (2006). Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis*, 1(3):515–533.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004). *Bayesian Data Analysis*. Chapman & Hall/CRC, second edition.

Geweke, J. (1989). Bayesian inference in econometric models using Monte Carlo integration. *Econometrica*, 57(6):721–741.

Gibbs, M. N. and Mackay, D. J. C. (2000). Variational Gaussian process classifiers. *IEEE Transactions on Neural Networks*, 11(6):1458–1464.

Gilks, W., Richardson, S., and Spiegelhalter, D. (1996). *Markov Chain Monte Carlo in Practice*. Chapman & Hall.

Gneiting, T. (1999). Correlation functions for atmospheric data analysis. *Quarterly Journal of the Royal Meteorological Society*, 125:2449–2464.

Gneiting, T. (2002). Compactly supported correlation functions. *Journal of Multivariate Analysis*, 83:493–508.

Goel, P. K. and Degroot, M. H. (1981). Information about hyperparamters in hierarchical models. *Journal of the American Statistical Association*, 76(373):140–147.

Good, I. J. (1952). Rational decisions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 14(1):107–114.

Grewal, M. S. and Andrews, A. P. (2001). *Kalman Filtering: Theory and Practice Using Matlab*. Wiley Interscience, second edition.

Harville, D. A. (1997). *Matrix Algebra From a Statistician's Perspective*. Springer-Verlag.

Kass, R. E. and Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795.

Kaufman, C. G., Schervish, M. J., and Nychka, D. W. (2008). Covariance tapering for likelihood-based estimation in large spatial data sets. *Journal of the American Statistical Association*, 103(484):1545–1555.

Kuss, M. and Rasmussen, C. E. (2005). Assessing approximate inference for binary Gaussian process classification. *Journal of Machine Learning Research*, 6:1679–1704.

Lawrence, N. (2007). Learning for larger datasets with the Gaussian process latent variable model. In Meila, M. and Shen, X., editors, *Proceedings of the Eleventh International Workshop on Artificial Intelligence and Statistics*. Omnipress.

Lawson, A. B. (2001). *Statistical Methods in Spatial Epidemology*. John Wiley & Sons, Ltd.

Matheron, G. (1973). The intrinsic random functions and their applications. *Advances in Applied Probability*, 5(3):439–468.

Minka, T. (2001). *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology.

Møller, J., Syversveen, A. R., and Waagepetersen, R. P. (1998). Log Gaussian Cox processes. *Scandinavian Journal of Statistics*, 25:451–482.

Moreaux, G. (2008). Compactly supported radial covariance functions. *Journal of Geodecy*, 82(7):431–443.

Neal, R. (1998). Regression and classification using Gaussian process priors. In Bernardo, J. M., Berger, J. O., David, A. P., and Smith, A. P. M., editors, *Bayesian Statistics 6*, pages 475–501. Oxford University Press.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer.

Neal, R. M. (1997). Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification. Technical Report 9702, Dept. of statistics and Dept. of Computer Science, University of Toronto.

Neal, R. M. (2003). Slice sampling. *The Annals of Statistics*, 31(3):705–767.

Nickisch, H. and Rasmussen, C. E. (2008). Approximations for binary Gaussian process classification. *Journal of Machine Learning Research*, 9:2035–2078.

O'Hagan, A. (1978). Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society. Series B.*, 40(1):1–42.

O'Hagan, A. (1979). On outlier rejection phenomena in Bayes inference. *Journal of the Royal Statistical Society. Series B.*, 41(3):358–367.

O'Hagan, A. (2004). Dicing with the unknown. *Significance*, 1:132–133.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(3):1939–1959.

Rasmussen, C. E. (1996). *Evaluations of Gaussian Processes and Other Methods for Non-linear Regression*. PhD thesis, University of Toronto.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Rathbun, S. L. and Cressie, N. (1994). Asymptotic properties of estimators for the parameters of spatial inhomogeneous poisson point processes. *Advances in Applied Probability*, 26(1):122–154.

Richardson, S. (2003). Spatial models in epidemiological applications. In Green, P. J., Hjort, N. L., and Richardson, S., editors, *Highly Structured Stochastic Systems*, pages 237–259. Oxford University Press.

Robert, C. P. and Casella, G. (2004). *Monte Carlo Statistical Methods*. Springer, second edition.

Rue, H., Martino, S., and Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal statistical Society B*, 71(2):1–35.

Sanchez, S. M. and Sanchez, P. J. (2005). Very large fractional factorials and central composite designs. *ACM Transactions on Modeling and Computer Simulation*, 15:362–377.

Sansò, F. and Schuh, W.-D. (1987). Finite covariance functions. *Journal of Geodesy*, 61(4):331–347.

Seeger, M. (2005). Expectation propagation for exponential families. Technical report, Max Planck Institute for Biological Cybernetics, Tübingen, Germany.

Seeger, M. (2008). Bayesian inference and optimal design for the sparse linear model. *Journal of Machine Learning Research*, 9:759–813.

Seeger, M., Williams, C. K. I., and Lawrence, N. (2003). Fast forward selection to speed up sparse Gaussian process regression. In Bishop, C. M. and Frey, B. J., editors, *Ninth International Workshop on Artificial Intelligence and Statistics*. Society for Artificial Intelligence and Statistics.

Snelson, E. (2007). *Flexible and Efficient Gaussian Process Models for Machine Learning*. PhD thesis, University College London.

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian process using pseudo-inputs. In Weiss, Y., Schölkopf, B., and Platt, J., editors, *Advances in Neural Information Processing Systems 18*. The MIT Press.

Snelson, E. and Ghahramani, Z. (2007). Local and global sparse Gaussian process approximations. In Meila, M. and Shen, X., editors, *Artificial Intelligence and Statistics 11*. Omnipress.

Spiegelhalter, D. J., Best, N. G., Carlin, B. P., and van der Linde, A. (2002). Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society B*, 64(4):583–639.

Storkey, A. (1999). *Efficient Covariance Matrix Methods for Bayesian Gaussian Processes and Hopfield Neural Networks*. PhD thesis, University of London.

Sundararajan, S. and Keerthi, S. S. (2001). Predictive approaches for choosing hyperparameters in gaussian processes. *Neural Computation*, 13(5):1103–1118.

Takahashi, K., Fagan, J., and Chen, M.-S. (1973). Formation of a sparse bus impedance matrix and its application to short circuit study. In *Power Industry Computer Application Conference Proceedings*. IEEE Power Engineering Society.

Tierney, L. and Kadane, J. B. (1986). Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, 81(393):82–86.

Titsias, M. K. (2009). Variational learning of inducing variables in sparse Gaussian processes. *JMLR Workshop and Conference Proceedings*, 5:567–574.

Tokdar, S. T. and Ghosh, J. K. (2007). Posterior consistency of logistic Gaussian process priors in density estimation. *Journal of Statistical Planning and Inference*, 137:34–42.

Vanhatalo, J., Pietiläinen, V., and Vehtari, A. (2010). Approximate inference for disease mapping with sparse gaussian processes. *Statistics in Medicine*, 29(15):1580–1607.

Vanhatalo, J. and Vehtari, A. (2007). Sparse log Gaussian processes via MCMC for spatial epidemiology. *JMLR Workshop and Conference Proceedings*, 1:73–89.

Vanhatalo, J. and Vehtari, A. (2008). Modelling local and global phenomena with sparse Gaussian processes. In McAllester, D. A. and Myllymäki, P., editors, *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 571–578.

Vanhatalo, J. and Vehtari, A. (2010). Speeding up the binary Gaussian process classification. In Grünwald, P. and Spirtes, P., editors, *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 1–9.

Vehtari, A. and Lampinen, J. (2002). Bayesian model assessment and comparison using cross-validation predictive densities. *Neural Computation*, 14(10):2439–2468.

Wendland, H. (1995). Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4(1):389–396.

Wendland, H. (2005). *Scattered Data Approximation*. Cambridge University Press.

West, M. (1984). Outlier models and prior distributions in Bayesian linear regression. *Journal of the Royal Statistical Society. Series B.*, 46(3):431–439.

Wiener, N. (1949). *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. MIT Press.

Williams, C. K. I. (1996). Computing with infinite networks. In Mozer, M. C., Jordan, M. I., and Petsche, T., editors, *Advances in Neural Information Processing Systems*, volume 9. The MIT Press.

Williams, C. K. I. and Barber, D. (1998). Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351.

Williams, C. K. I. and Rasmussen, C. E. (1996). Gaussian processes for regression. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 514–520. MIT Press.

Wu, Z. (1995). Compactly supported positive definite radial functions. *Advances in Computational Mathematics*, 4(1):283–292.